
Universitat Oberta de Catalunya

Doctoral Thesis

# Efficient Modeling of High Order Functions in Computer Vision

*Marc Oliu Simón*

supervised by
Dr. Xavier Baró Solé
Dr. Sergio Escalera Guerrero

May 10, 2022

# *Acknowledgements*

I would first like to thank my advisors, Xavier Baró and Sergio Escalera, for their guidance and, above all, patience during all these years. It has been a long journey, with periods of hard and fruitful work and others where things just wouldn't work out as we'd all have liked, but they've always been there, whether to help push the thesis forward or eventually steer me away from failures and towards new directions.

Also thanks to all those with which I have collaborated. To Stan Sclaroff and Sarah Adel Bargal, for having me at Boston University for my research stay and for guiding me on the research I did while there. To Isabelle Guyon and Hugo Jair Escalante, with which we worked together to organize various workshops, and to Jeffrey Cohn, Takeo Kanade and Thomas B. Moeslund, for the chance at collaborating and publishing together.

A special thanks to the many colleagues I made during the PhD. We have suffered together, but also celebrated; Carles, Albert, Ciprian, Dani, Javier, Alejandro, Cristina, Camilo, Pablo, Julio, Hugo, and many others. Without others with which to share ideas, coffees, beers, and especially vent frustrations, many of us would have lost it along the way.

I finalment gràcies a la meva família i amics. Al meu pare Xevi, a la Carme, a la meva mare Marta i a en Carlos, per donar-me suport quan ha sigut necessari, tot i no acabar d'entendre ben bé què és això que faig. A en Joan i la Marina, que han sigut sempre presents i amb qui de manera inesperada he compartit l'estança.

# Abstract

Recent years have seen the field of Computer Vision advance by leaps and bounds, a phenomenon related to the advent of compute GPUs. With these, techniques such as deep learning have become common place, with other related approaches such as adversarial networks coming to be staples of machine learning and more specifically Computer Vision. While this has resulted in a fast pace of improvement to state-of-the-art results across the board, it has been at the expense of much larger memory usage and computational cost, vastly increasing the hardware requirements for training and in many cases executing such methods. Techniques for mitigating this problem are mostly focused on the post-training stage, where an already trained model is iteratively trimmed and optimized in order to reduce its total number of parameters.

In this thesis we explore the usage of high order methods to reduce this computational cost and memory usage, both for classical approaches and in the case of neural networks. Instead of trying to reduce the size and cost of an already trained method, we aim to reduce them by design, which results not only in a faster and more compact trained model, but also a smaller memory footprint and computational cost during training. Any additional distillation methods could still be applied afterwards. Although the proposed methods are generic, and thus applicable to a wide array of problems, this thesis is centered around Computer Vision, and more specifically Human pose and behavior analysis.

Three different but related approaches are presented for such a task. Firstly, we introduce a simple but effective cascaded regression approach, demonstrating the effectiveness of higher order regression in reducing the memory and computational needs of this already classical approach, while also increasing the robustness of the method. Secondly, we propose a new type of recurrent auto-encoder for video prediction that directly shares its memory states between the encoder and decoder. This simple change can be exploited in a variety of ways, resulting in a much lower resource usage, thanks to its ability to more efficiently leverage the higher order dynamics of a sequence. Finally, we propose a domain adaptation method that achieves results comparable to those of the commonly used adversarial approaches by implicitly aligning high order estimates of both domains probability density functions. This is done in such a way that no additional parameters or training steps are necessary, easily outperforming said adversarial approaches in terms of resource usage.

# Resum

En els darrers anys hem vist el camp de la Visió per Computador avançar ràpidament, un fenomen relacionat amb l'advent de les GPU de còmput. Amb aquestes, tècniques tal com l'aprenentatge profund han esdevingut populars, amb altres tècniques relacionades, com ara xarxes adversàries, convertint-se en eines freqüentment utilitzades en l'aprenentatge per ordinador i, més precisament, Visió per Computador. Mentre que això ha resultat en un ràpid avenç de l'estat de l'art en tots els àmbits, aquest ha vingut de la mà d'un major requeriment en memòria i cost computacional, incrementant els requisits de maquinari per entrenar i en molts casos executar aquests mètodes. Les tècniques existents per mitigar aquest problema s'enfoquen principalment en l'etapa de postentrenament, on un model ja entrenat es poda i optimitza iterativament per tal de reduir el nombre total de paràmetres.

En aquesta tesi explorem l'ús de mètodes d'alt ordre per tal de reduir l'ús de memòria i cost computacional, tant en tècniques clàssiques com en xarxes neuronals. En comptes d'intentar de reduir la mida i cost d'un model ja entrenat, el nostre objectiu és de reduir-los per disseny, el qual resulta no només en un model entrenat més ràpid i compacte, però també en uns requeriments computacionals i de memòria menors durant l'entrenament. Altres mètodes per destil·lar la xarxa segueixen podent-se aplicar posteriorment. Encara que els mètodes proposats són genèrics i, per tant, aplicables a una ampla gamma de problemes, aquesta tesi se centra en Visió per Computador i, més concretament, en l'anàlisi de postura i comportament humans.

Per aquesta tasca es proposen tres mètodes diferents però relacionats. Primerament, introduïm un mètode de regressió en cascada simple però efectiu, demostrant la utilitat dels regressors d'alt ordre en reduir la memòria i cost computacional d'aquesta tècnica clàssica, incrementant la robustesa del mètode de manera simultània. En segon lloc, proposem un nou tipus de codificador automàtic per a la predicció de vídeo que directament comparteix els estats de memòria entre el codificador i descodificador. Aquest simple canvi es pot explotar en una gran varietat de maneres, resultant en un ús de recursos molt més baix gràcies a una codificació més eficient de les dinàmiques d'alt ordre de les seqüències. Finalment, proposem un mètode d'adaptació de domini que obté resultats similars als obtinguts per les tècniques adversàries més freqüentment utilitzades, mitjançant l'alineació d'estimacions d'alt ordre de les funcions de densitat de probabilitat. Això ho fem de tal manera que no es necessiten paràmetres o passos d'entrenament addicionals, reduint l'ús de recursos significativament comparat amb les tècniques adversàries.

# Resumen

En los últimos años, hemos visto el campo de la Visión por Computador avanzar rápidamente, un fenómeno relacionado con el advenimiento de las GPU de cómputo. Con ellas, técnicas como el aprendizaje profundo se han vuelto populares, mientras que otras técnicas relacionadas, como las redes adversarias, se han convertido en herramientas frecuentemente utilizadas en el aprendizaje por ordenador y, más específicamente, Visión por Computador. Mientras que eso ha resultado en un rápido avance del estado del arte en todos los ámbitos, eso ha venido de la mano de un mayor requerimiento en memoria y coste computacional, incrementando los requisitos de maquinaria para entrenar y en muchos casos ejecutar dichos métodos. Las técnicas existentes para mitigar el problema se enfocan principalmente en la etapa post-entrenamiento, donde un modelo ya entrenado se poda y optimiza iterativamente con tal de reducir el número total de parámetros.

En esta tesis exploramos el uso de métodos de alto orden con tal de reducir el uso de memoria y coste computacional, tanto en técnicas clásicas como en redes neuronales. En lugar de intentar reducir el tamaño y coste de un modelo ya entrenado, nuestro objetivo es reducirlos por diseño, lo cual resulta no solo en un modelo entrenado más rápido y compacto, sino que también en unos requerimientos computacionales y de memoria menores durante el entrenamiento. Otros métodos para destilar la red siguen pudiéndose aplicar posteriormente. Aunque los métodos propuestos son genéricos y, por tanto, aplicables a un gran rango de problemas, esta tesis se centra en Visión por Computador y, más concretamente, en el análisis de la pose y comportamiento humanos.

Para esta tarea se plantean tres métodos diferentes pero relacionados. Primero introducimos un método de regresión en cascada simple pero efectivo, demostrando la utilidad de los regresores de alto orden en reducir la memoria y coste computacional de esta técnica clásica y, simultáneamente, incrementando la robustez del método. En segundo lugar, introducimos un nuevo tipo de codificador automático para la predicción de video que directamente comparte los estados de memoria entre el codificador y decodificador. Este simple cambio se puede explotar en una gran variedad de formas, resultando en un uso de recursos muy inferior gracias a una codificación más eficiente de las dinámicas de alto orden en las secuencias. Finalmente, proponemos un método de adaptación de dominio que obtiene resultados similares a los obtenidos por las más comúnmente utilizadas técnicas adversarias, utilizando la alineación de estimaciones de alto orden de las funciones de densidad de probabilidad. Eso lo hacemos de tal modo que no se necesitan parámetros o pasos de entrenamiento adicionales, reduciendo el uso de recursos significativamente comparado con las técnicas adversarias.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The ability to make sense of visual stimuli was once thought of as something inherent to human beings, and to a lesser degree other animals. While computers were capable of extensive computation, far surpassing the ability of any one individual by orders of magnitude, it was thought that such visual tasks such as face recognition, the identification of facial expressions, recognition of objects and the interaction between different elements in a scene would forever remain the domain of humans.

This started to change decades ago, with the introduction of some now classical machine learning approaches that could, in a limited fashion, extract knowledge from images. With the later arrival of massively parallel computing and subsequent development of ever deeper neural networks trained on that hardware, the picture quickly changed. In less than a decade, Computer Vision went from a research domain barely applicable to simple problems, to matching and in some cases even surpassing humans in a wide array of visual recognition tasks. Face alignment [1, 2, 3], facial expression recognition [4, 5], object recognition [6, 7], human pose recovery [8], and even more complex tasks such as scene understanding [9, 10] and future video prediction [11, 12, 13] are just a few examples.

## 1.1 Motivation

The explosive growth of Computer Vision and the attainment of ever higher accuracies in a wide variety of problems is in no small part due to the development of deep learning approaches with a higher number of hidden layers and parameters. The first such successful topology was LeNet [14], introduced in 1989 by Y. LeCun. The combination of a deep topology, the use of convolutional layers and its training through back-propagation made it the foundation of modern Computer Vision.

One advantage to CNNs, and the reason why deep learning later became so popular in the field, is the implicit regularization brought by its convolutional topology. Each pixel in an image is processed by the same convolutional kernel. This results on the same set of parameters having a number of effective samples equal to the resolution of an image, for each image in the batch. This, combined with the smaller number of parameters on each layer when compared to a fully connected topology, means that the network can be trained with a smaller number of samples.

It wasn't until much later, though, that deep convolutional networks garnered

popularity. While the results by Y. LeCun were impressive, the main problem with the algorithm was one of computational cost: training such a network at the time required CPU super-computers and lengthy training periods, making it difficult and expensive for other teams to replicate his results or apply the same method to other datasets. It was in 2012, with the publication of the Alexnet topology introduced by A. Krizhevsky *et al.* [15], that this kind of approach garnered interest. The new topology had 8 layers in total, one more than the LeNet, and a much larger number of trainable parameters per layer.

What made it possible to train inexpensively was its use of Graphics Processing Units (GPU) for both forward and back-propagation, both neural models and GPU hardware being well suited for massively parallel computing. In other words, by then the hardware technology had finally caught up with the computational requirements of such models. It took a single desktop computer with two GPUs less than six days to fully train the model. Another reason for its success was the replacement of classical activation functions such as the hyperbolic tangent and sigmoid functions with Rectifier Linear Units (ReLU), a type of linear activation truncated at zero for negative values. This was a type of activation function initially proposed in 2010 for Restricted Boltzmann Machines [16] and first introduced to back-propagation based training in this work. It has the advantage of being both a less expensive and non-saturating function. This mitigates the problem with vanishing gradients [17], where gradients tend to zero during back-propagation as the topology gets deeper.

Since then, many of the gains in accuracy have come from increasing the depth of such networks, enabled by ever more powerful (and costly) GPUs. VGG-16 [18] followed in 2014 with 16 layers, GoogLeNet [19] in 2015 with 22 layers, and ResNet [20] in 2016 with 152 layers. ResNet models proposed the use of residual connections, creating shortcuts between layers that produced shorter paths between input and output, bypassing the re-emerging vanishing gradients problem to a large degree and thus allowing for much deeper topologies.

Still, such models require ever larger amounts of memory and compute power, the issue magnified when considering temporal models working on sequences such as Recurrent networks [21, 22], and techniques trying to model one or multiple domains as a whole, using a multi-step training approach, as is the case of Generative Adversarial Networks (GAN) [23]. In this work, we leverage high order methods to try and reduce both memory and computational cost, both explicitly for classical machine learning, and in an implicit but clear fashion for deep learning, where either the topology or an auxiliary loss are used to more efficiently model high order relationships within the data.

## 1.2 Contributions

This thesis approaches Human pose and behavior analysis with the goal of introducing more efficient Computer Vision techniques in terms of both memory and computational cost. This is achieved through the analysis and modeling of high order relationships in the data, providing methods that more efficiently leverage these relationships. We approach three different sub-tasks to do so: Facial landmark alignment as a recognition task, future video prediction as a human behavior analysis task, and domain adaptation as a method for leveraging unlabeled data. These three approaches each provide a different solution to reducing the model

complexity, as summarized below.

### 1.2.1 Second order linear methods

We first study the use of multivariate polynomial regressors as a direct approach to modeling higher-order relationships between features in a classical machine learning approach: linear regression. We specifically focus on face alignment, the process of finding a subset of facial landmarks, usually corresponding to salient features such as the contour of the mouth and eyes, the nose, and profile of the face. Classically, this is achieved through cascaded regression, where a series of linear regressors iteratively refine an initial guess to the landmark locations [1, 24].

The problem with these approaches is their inherent inability to tackle high-variability scenarios, such as highly rotated faces or significant changes in illumination. Usually, this is solved by creating an ensemble of regressors at each stage, each one trained on a subset of angles and illuminations, and resulting in a significant increase in the total number of parameters. Polynomial regression was never before considered due to it usually resulting in a quadratic increase on the number of parameters, where $N$ initial features would result in $N(N+1)+1$ parameters per target variable instead of the $N+1$ of a simple linear regressor.

We solve this problem by considering the variance distribution on the data, highly limiting the number of second-order interactions between features. This in turn heavily reduces the total number of parameters, bringing it down to below the amount required for an ensemble model. At the same time, the robustness of the final model exceeds that of the ensemble-based approaches.

### 1.2.2 Shared-state Neural Networks

Moving on to deep learning, and more specifically to recurrent generative models, we introduce a new topology for future video prediction that better leverages the high order relationships within the data.

Future video prediction is the task of generating the following frames in a video sequence given the ones immediately before them. This task is generally solved using some kind of recurrent convolutional model, consisting of both an encoder and a decoder. The general pipeline of such a model is as follows: first, a series of known frames are passed through the network. These initialize the states of both encoder and decoder with the known video sequence. Afterwards, frames are generated using the decoder and fed back to the encoder, creating a feedback loop that keeps producing frames past the end of the sequence, effectively predicting its continuation.

This type of topology has some glaring weaknesses, especially when considering it through the point of view of high order interactions. In a neural network, each layer is capable of doubling the maximum order of the considered relationships between variables (see chapter 3). The first layer will only consider linear relationships, regardless of the subsequent activation function. The second layer will be able to capture quadratic relationships, while the third will potentially be capturing fourth-order relations. The problem is that, in a linear pipeline where the input is fed to the first layer of the encoder and the output is produced by the last one of the decoder, each layer must capture an (over-)complete representation of the input, forcing the layers capable of encoding the higher order relations to also maintain a representation of the lower level ones.

We solve this problem with a new type of recurrent auto-encoder that fully shares its states between the encoder and corresponding layers of the decoder. We show how such a model, apart from directly halving the needed memory and computational cost, it additionally frees the higher order layers so that they can focus on capturing the high order dynamics instead of the full sequence, further reducing the number of required parameters. It also allows us to remove layers from the network after training, without needing to retrain the network. This means that we can adjust the complexity of the model for free after training based on the order of feature relationships present in the data.

### 1.2.3 Cumulative Distribution Estimation

For our third contribution, we develop a Domain Adaptation (DA) method based on the estimation of the Probability Distribution Function (PDF), indirectly performing a high order modeling of the same.

Domain Adaptation is the task of matching and aligning two similar probability distributions, so that the discrepancies between two sets of data are not present in their feature representation. This comes in handy for a variety of tasks, the most prominent of which is leveraging unlabeled data when training a model. In such a case, the labeled and unlabeled data, which might come from different datasets, are considered as two different domains. A neural network is then trained to predict the labeled data, and at the same time minimize the discrepancies between both domains in the feature space.

Two families of methods exist for such a task. The first models the distributions through simple statistical approximations, such as multi-variate Gaussians. While such approaches work, they completely ignore the internal structure of the distributions, effectively using a second-order approximation of the distribution, and might end up incorrectly aligning them. The second family uses a secondary non-linear model to approximate the distributions, such as an additional neural network (most commonly a GAN) [25, 26, 27, 28]. These models successfully capture the complexity of the distribution by encoding the high order relationships of the samples into additional adversarial discriminators, but do so at the cost of additional trainable parameters, memory and computational cost, often further introducing an additional adversarial training step.

We propose an approach more related to the classical statistical distribution approximation methods, but capable of modeling the high order statistical distribution of the samples. Furthermore, it does so without the need for learnable parameters and with only a minimal amount of additional memory.

## 1.3 Publications

The following publications are part of this thesis, either in a direct or indirect fashion. Grayed out entries correspond to works under review as of the time of this writing.

### 1.3.1 Journal papers

- *M. Oliu, C.A. Corneanu, K. Nasrollahi, O. Nikisins, S. Escalera, Y. Sun, H. Li, Z, Sun, T.B. Moeslund, M. Greitans.* (2016) **Improved RGB-DT**

**based face recognition**. IET Biometrics.

- *C.A. Corneanu, M. Oliu, J.F. Cohn, S. Escalera.* (2016) **Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-related Applications**. Transactions on Pattern Analysis and Machine Intelligence.

### 1.3.2 International conferences and workshops

- *I. Ramin, K. Nasrollahi, M. Oliu, C.A. Corneanu, S. Escalera, C. Bahnsen, D.H. Lundtoft.* (2015) **Spatiotemporal analysis of RGB-DT facial images for multimodal pain level recognition**. Computer Vision and Pattern Recognition Workshops.

- *S. Escalera, J. Gonzalez, X. Baro, P. Pardo, J. Fabian, M. Oliu, H.J. Escalante, I. Huerta, I. Guyon.* (2015) **Chalearn looking at people 2015 new competitions: Age estimation and cultural event recognition**. International Joint Conference on Neural Networks.

- *S. Escalera, M. Torres, B. Martinez, X. Baro, H.J. Escalante, I. Guyon, G. Tzimiropoulos, C. Corneanu, M. Oliu, M. Ali.* (2016) **Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016**. Computer Vision and Pattern Recognition Workshops.

- *S. Escalera, M. Torres, B. Martinez, X. Baro, H.J. Escalante, I. Guyon, G. Tzimiropoulos, C. Corneanu, M. Oliu, M. Ali.* (2016) **Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016**. Computer Vision and Pattern Recognition Workshops.

- *B. Chen, S. Escalera, I. Guyon, V. Ponce, N. Shah, M. Oliu.* (2016) **Overcoming Calibration Problems in Pattern Labeling with Pairwise Ratings: Application to Personality Traits**. European Conference on Computer Vision Workshops.

- *V. Ponce, B. Chen, M. Oliu, C. Corneanu, A. Clapes, I. Guyon, S. Escalera.* (2016) **Chalearn lap 2016: First round challenge on first impressions-dataset and results**. European Conference on Computer Vision Workshops.

- *M. Oliu, C.A. Corneanu, L.A. Jeni, J.F. Cohn, T. Kanade, S. Escalera.* (2016) **Continuous Supervised Descent Method for Facial Landmark Localization**. Asian Conference on Computer Vision.

- *M. Oliu, J. Selva, S. Escalera.* (2018) **Folded Recurrent Neural Networks for Future Video Prediction**. European Conference on Computer Vision.

- *D. Sanchez, M. Oliu, M. Madadi, X. Baró, S. Escalera.* (2019) **Multi-task human analysis in still images: 2D/3D pose, depth map, and multi-part segmentation**. International Conference on Automatic Face & Gesture Recognition.

- *M. Oliu, S.A. Bargal, X. Baró, S.escalera* (2022) **Multi-varied Cumulative Alignment for Domain Adaptation**. International Conference on Image Analysis and Processing.

## 1.4 Thesis outline

The thesis is structured around three main parts, each introducing a different machine learning approach to the efficient modeling of high order functions, both in terms of memory and computational requirements. All parts share a similar structure, with a specific problem being introduced, its related literature explained and a specific solution being put forward for said problem, followed by experiments and conclusions.

Part I introduces a simple strategy for cascaded linear regression approaches through the efficient use of multivariate quadratic regression, resulting in much more expressive, yet compact models.

Following that, Part II delves into deep learning. First, the relationship between high order regressors and the depth of neural network models is established in Chapter 3. This information is then leveraged in Section 4 to improve the flow of information in recurrent auto-encoders, greatly reducing both the memory and computational requirements.

Part III is focused on a new metric for Domain Adaptation, where highly complex probability distributions can be aligned while bypassing the need to directly model the distributions, again resulting in a more compact model.

# Part I

# Second order linear methods

# Chapter 2

# Continuous Supervised Descent Method

## 2.1 Introduction

Facial landmark localization consists of detecting a set of particular points on the face. Usually these points have semantic meaning, their location being in highly distinctive places around the eyes, mouth or nose. A set of such points is useful for expressing both the rigid and non-rigid deformations of the face geometry. Because facial geometry changes with identity, facial expression and head pose, it is an important step in many automatic facial analysis tasks such as face recognition, face expression recognition, face synthesis and age or gender estimation [4].

A common approach for locating landmarks on the face is to model the relation between the face appearance and its geometry. If we consider $\mathbf{X}^*$ to be the ground truth geometry, and $\Phi(\mathbf{I}, \mathbf{X})$ a representation function of a geometry $\mathbf{X}$ on an image $\mathbf{I}$, then starting from an initial estimation $\mathbf{X}^0$ landmark location can be formulated as an optimisation problem of the form:

$$\arg\min_{\Delta\mathbf{X}} f(\mathbf{X}^0 + \Delta\mathbf{X}) = ||\Phi(\mathbf{I}, \mathbf{X}^0 + \Delta\mathbf{X}) - \Phi(\mathbf{I}, \mathbf{X}^*)||_2^2 \tag{2.1}$$

Because $\Phi$ is a highly non-linear function, $f$ is non-convex and has many local minima, the problem becomes severe in the case of large variations of the texture which is normally the case with rotations of the head and strong non-rigid deformations. Additionally, successfully solving the optimisation problem is highly dependent on the initialisation.

Historically, Active appearance models (AAM) [29] are one of the most used methods for 2D face registration. They are an extension of active shape models (ASM) [30] which encode both geometry and intensity information. More recently, even though single step landmark location methods have been proposed [31, 32], the most common approach is to model the relationship between texture and geometry with a cascade of regression functions [2, 33, 34, 1, 24, 35, 36]. Features are extracted from the current estimated geometry and passed to the learnt mapping in order to update the geometry. This process is repeated iteratively for each step of the cascade, applying a specific mapping to each. If we denote by $\mathbf{R}^i$ the regression function at the $i$th step of the cascade, by $\mathbf{\Phi}^i = \Phi(\mathbf{I}, \mathbf{X}^i)$ the corresponding

representation and by $\mathbf{b}^i$ a constant bias, then at every step of the cascade, the geometry $\mathbf{X}$ will be updated in the following way:

$$\mathbf{X}^{i+1} = \mathbf{X}^i + \mathbf{R}^i \mathbf{\Phi}^i + \mathbf{b}^i \tag{2.2}$$

While most cascaded regression methods share this approach, considerable variation can be found in representation, regression functions and initialisation strategies. The simplest way to initialise the geometry is by starting with the mean [1, 35, 34]. For faces, this works well in close-to-frontal scenarios but proves inefficient when large pose variation occurs. A common solution is to try a set of random initialisations and consider the median of the predictions as the final solution [37, 2]. Unfortunately, this considerably increases the computational cost. An alternative approach is to apply the initial part of the cascade and continue only if the variance of the regressed shapes is low, which is a strong predictor of convergence towards the global minimum [33]. If this is not the case then a different set of initial shapes is generated. Even so, all these methods are dependent on the initialisation and prove low generalisation to large head pose rotation. A coarse-to-fine searching approach was recently proposed to deal with the initialisation dependency problem [3]. A regression function is learnt from a set of shapes generated according to a probabilistic distribution on the shape space. A dominant set approach is used to eliminate outliers between the regressed shapes in an unsupervised manner. From the filtered subset the centre of a smaller region of the original space is computed and the process repeated until convergence. While it prevents locality of the solutions it improves robustness to large pose variation.

The work of Dollar et al. which proved influential in the field of facial landmark localization, uses intensities of sparse sets of pixels at predefined locations to represent texture in a shape indexed fashion for learning a fixed linear sequence of weak regressors [37]. In this way, representation's output depends on both the image data and the current estimate of the geometry. Some of the methods propose to jointly learn the representation and the regression function [2, 33, 35, 34]. In this sense, several shape indexed locations are randomly generated and then selected based on a certain optimization criteria. Alternatively, local binary features are learnt for each landmark independently [34]. During test, very fast landmark localization is obtained. In a recent method [38], *Difference of Gaussians* (DoG) features are selectively extracted from locations arranged in a pattern inspired by the human visual system [39]. Learnt trees at early stages tend to select indexed DoG features computed from distant sampling points while trees at later stages tend to use nearby sampling points. Finally, a very common problem of most of the proposed methods, the lack of sensitivity to occlusions is tackled in the work of Burgos et al. [33]. They propose a method that reduces exposure to outliers by detecting occlusions explicitly and using robust shape-indexed features. It incorporates occlusion directly during learning to improve shape estimation.

A distinct group of methods use predefined handcrafted representations while learning the regression function from the data. For example, to overcome the large computational time required by the regression of many generated shapes at each stage more simple descriptors are used in the initial stages when coarse localisation is performed. More complicated representations are used on final stages when fine localisation takes place [3]. A particularly important set of methods that use fixed representations are the ones derived from the *Supervised Descent Method* (SDM) [1]. SDM uses simplified SIFT features and linear regressors. As is the

case of previous methods, SDM works well for near frontal faces but fails on strong rotations. To overcome this problem, *Global Supervised Descent Method* (GSDM) [24] introduced an approach which uses a sub-space defined by a set of directions of maximum variance of the training data to partition the original feature space. Each partition shares a similar descent direction for the training instances falling within it. A linear regressor is learnt for each partition. However, GSDM suffers from two main problems. Both the number of training instances and model size increase exponentially with the number of sub-space dimensions.

In order to perform landmark localisation under strong rotations while keeping a fast and compact model, this work proposes a continuous formulation of GSDM. Instead of using the sub-space to partition the feature space as GSDM does, it is used to describe a space of linear regressors. This is equivalent to proposing a regressor which estimates the second derivative of the gradient, instead of the first as a standard linear regressor would (e.g. in SDM). While this formulation may not be as expressive as GSDM, the amount of memory and training instances required increases linearly with the number of dimensions of the sub-space. Also, the proposed formulation defines a specific linear regressor for each instance.

In summary, our list of contributions is as follows:

- we present a method that improves state-of-the-art results on strongly rotated faces
- the trained models are small, the amount of memory and training instances required increase only linearly with the number of dimensions of the sub-spaces
- the method is fast to train due to its closed form solution
- we have synthesised largest 2D face dataset to date, with a challenging face rotation distribution

The rest of this paper is organised as follows: in Section 2.2 we formulate the proposed method, in Section 2.3 we present the experimental analysis and finally, in Section 2.4, we conclude the paper.

**Notations.** Vectors ($\mathbf{a}$) and matrices ($\mathbf{A}$) are denoted by bold letters. An $\mathbf{u} \in \mathbb{R}^d$ vector's Euclidean norm is $\|\mathbf{u}\|_2 = \sqrt{\sum_{i=1}^{d} u_i^2}$. $\mathbf{B} = [\mathbf{A}_1; \ldots; \mathbf{A}_K] \in \mathbb{R}^{(d_1+\ldots+d_K)\times N}$ denotes the concatenation of matrices $\mathbf{A}_k \in \mathbb{R}^{d_k \times N}$.

## 2.2 Continuous Supervised Descent Method

### 2.2.1 Second order regressor

The original SDM method [1] is an exemplar-based method which learns a series of linear regressors approximating the data to the global optima in a cascaded manner. Lets consider $\mathbf{X}^i \in \mathbb{R}^{n\times m}$ the $m$ targets for each of $n$ samples at a given cascade step $i$, $\Delta\mathbf{\Phi}^i \in \mathbb{R}^{n\times(k+1)} = \mathbf{\Phi}^i - \overline{\mathbf{\Phi}^i}$ the difference of the feature vectors of length $k$ from the mean, with a column vector of ones added in order to account for the bias, and $\mathbf{R}^i \in \mathbb{R}^{(k+1)\times m}$ the linear regressor for each of the $m$ parameters. Then the update formula for SDM can be expressed as follows:

$$\mathbf{X}^{i+1} = \mathbf{X}^i + (\mathbf{\Phi}^i - \overline{\mathbf{\Phi}^i})\mathbf{R}^i = \mathbf{X}^i + \Delta\mathbf{\Phi}^i\mathbf{R}^i \tag{2.3}$$

This can be seen as learning a linear approximation of the first-order partial derivatives for each parameter. These correspond to $\partial \Delta \mathbf{X}^{i+1} / \partial \Delta \mathbf{\Phi}^i_j = \Delta \mathbf{\Phi}^i_j \mathbf{R}^i_j$, with $\mathbf{R}^i$ being the Jacobian matrix, $\Delta \mathbf{\Phi}^i_j$ the $j$th column of $\Delta \mathbf{\Phi}^i$ and $\mathbf{R}^i_j$ the $j$th row of $\mathbf{R}^i$. To make this approximation, the slope is considered homogeneous for any point of the feature space. This assumption does not hold for most problems, where the gradient direction suffers from large variations on different locations of that space. On Global SDM [24] these variations are handled by partitioning the space into different regions and learning a linear regressor for each one. This approach can approximate with high accuracy the gradient variations at different regions of the space, but has the problem of doubling the amount of learnt regressors and required training data each time the space is divided.

Here we introduce a continuous formulation, where a set of bases are learnt for the regressors, effectively learning a linear approximation of the second derivative. To do so, first a set of main modes of variation are learnt from either $\Delta \mathbf{X}^*$ or $\Delta \mathbf{\Phi}^i$ using Principal Component Analysis (PCA):

$$\Delta \widetilde{\mathbf{\Phi}^i} = \left[ \Delta \mathbf{\Phi}^i \mathbf{P}_{1:l}, \mathbf{1}_n \right], \tag{2.4}$$

Where $l$ represents the number of bases to learn and $\mathbf{P}_{1:l}$ is the projection matrix. $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ denotes an all-ones vector. Given that the total number of learnable parameters for one of $m$ targets equals $p = (k+1)(l+1)$, learning the second derivative for all parameters ($l = k$) would drastically increase the problem dimensionality. Estimating the second derivative on the $l$ main variation modes is a more treatable problem. Given one of the targets $\Delta \mathbf{X}^i_j \in \mathbb{R}^{n \times 1}$, its associated second order regressor is expressed as the solution to the following minimisation problem:

$$\underset{\mathbf{R}^i_j}{\arg\min} \, ||(\Delta \mathbf{\Phi}^i \circ (\Delta \widetilde{\mathbf{\Phi}^i} \mathbf{R}^i_j)) \mathbf{1}_{(k+1)} - \Delta \mathbf{X}^i_j||_2^2 \tag{2.5}$$

Here, $\mathbf{R}^i_j \in \mathbb{R}^{(l+1) \times (k+1)}$ is the set of $l$ bases (and baseline or bias regressor) describing the regressor for the $j$th target at the $i$th cascade step, and $\circ$ denotes the Hadamard product. Note that, according to Equation 2.7, this formulation learns a linear approximation to the second order partial derivatives $\partial^2 \Delta \mathbf{X}^{i+1}_j / (\partial \Delta \mathbf{\Phi}^i_p \, \partial \Delta \widetilde{\mathbf{\Phi}}^i_q) = \Delta \mathbf{\Phi}^i_p \Delta \widetilde{\mathbf{\Phi}}^i_q (\mathbf{R}^i_j)_{pq}$. Thus $\mathbf{R}^i_j$ corresponds to a compact version of the Hessian matrix for target $j$ at cascade step $i$, having the dimensionality of the feature space reduced before applying the second derivative. Equation 2.5 can be seen as a compact formulation defining a quadratic regressor for each target, which is known to be a linear problem, having a closed form solution. This minimisation problem can be expressed in a least squares form, providing a closed form solution, as follows:

$$\underset{\mathbf{R}^i_j}{\arg\min} \, ||(\Delta \widetilde{\mathbf{\Phi}^i} \odot \Delta \mathbf{\Phi}^i) vec(\mathbf{R}^{i\intercal}_j) - \Delta \mathbf{X}^i_j||_2^2 \tag{2.6}$$

Here $\odot$ denotes the Khatri–Rao product, considering each instance (row) on $\Delta \mathbf{\Phi}^i$ and $\Delta \widetilde{\mathbf{\Phi}^i}$ as a partition of the matrix, and $vec(\mathbf{R}^{i\intercal}_j) \in \mathbb{R}^{(kl+2) \times 1}$ is the vectorisation of the regressor bases. Thus, while the second derivative estimate is used for a subset of principal components, the regressor remains linear. This allows us to rapidly and directly find the optimal regression weights given the training instances. Note that this formulation could be extended to estimate higher order derivatives

by applying the Khatri–Rao product multiple times. At test time, the parameters are updated with the following equation:

$$\mathbf{X}_j^{i+1} = \mathbf{X}_j^i + (\Delta\mathbf{\Phi}^i \circ (\Delta\widetilde{\mathbf{\Phi}^i}\mathbf{R}_j^i))\mathbf{1}_{(k+1)} \tag{2.7}$$

This formula estimates the regressor weights and bias for the current value of the principal components $\widetilde{\mathbf{\Phi}^i}$, and applies it to the features. This is more memory-efficient than performing the Khatri-Rao product of $\Delta\mathbf{\Phi}^i$ and $\Delta\widetilde{\mathbf{\Phi}}^i$ and then performing a linear regression. The bias for the regressor bases is the baseline regressor for an instance with the mean value for the $l$ principal components (PCs) of the feature vector. Each of the $l$ regression bases in $\mathbf{R}^i$ corresponds to the second derivative estimate wrt. a given PC. Note that when $l = 0$ the model is a standard linear regressor. Thus, SDM can be seen as a special case of our method where the second derivative is not taken into account for any PC.

The proposed approach estimates a standard linear regressor for each instance given the coordinates of the features sub-space $\Delta\widetilde{\mathbf{\Phi}}^i$. Global SDM assigns the same one to all instances falling into a given region of the partitioned sub-space. Another advantage of this approach is that the number of parameters learnt $p$ at each cascade step increases linearly with the number of bases ($p = (k + 1)(l + 1)$). With Global SDM it increases quadratically ($p = (k+1)min(1, l^2)$). These two factors make the proposed approach both more compact in terms of memory and more accurate, as shown in Section 2.3.3. Because the regression space is continuous, the weights of the linear regressor are adapted to each instance, providing more flexibility to the model. During training, this also implies that for the proposed approach all the training data is available for each base of the sub-space, helping to reduce overfitting. GDSM distributes the data between quadrants, logarithmically reducing the available training data for each quadrant with the number of sub-space bases.

### 2.2.2 Implementation details

As discussed in Section 2.2.1, the second derivative of the feature space is calculated over the $l$ principal components. For this work, similarly to [1], a simplified SIFT descriptor is extracted from each landmark estimate. The descriptor has a fixed $32 \times 32$ window around the landmark, rotated according to the in-plane rotation of the current geometry relative to the mean facial shape. PCA is then applied in order to reduce its dimensionality. Thus, the feature vector for an instance $j$ at the cascade step $i$ is defined as $\mathbf{\Phi}_j^i = sift(\mathbf{I}_j, \mathbf{X}_j^i)^\intercal \mathbf{P}_{1:k}^i$, the $k$ principal components of the extracted SIFT descriptors. This implicitly provides the $l$ parameters for the regressor bases, being $\widetilde{\mathbf{\Phi}_j^i} = (\mathbf{\Phi}_j^i)_{1:l}$. The targets $\Delta\mathbf{X}^i$ are rotated in the same way as the descriptor windows in order to maintain a coherent update direction.

The feature vector length $k$ and number of regression bases $l$ may depend on the problem and are free parameters of the model. Still, there are two considerations to take into account. In a cascaded regression approach, the first steps of the cascade broadly approximate the face pose and general shape, while later steps tend to fine-tune the location of each landmark, working more locally. This implies that at the fist steps a smaller amount of the total descriptors variance may be enough. Conversely, a higher amount of regression bases would increase the adaptability to the descriptors main modes of variation, which are expected to be caused by pose/illumination variations. The feature vector length $k$ is defined as a fixed percentage of the original SIFT features variance. While it may be possible to adjust

the number of bases $l$ at each cascade step (for instance with forward selection), in this work a global value is chosen for all cascade steps.

The initial shape at the first cascade step is the mean shape. It is calculated from the training instances ground truth shapes using Generalised Procrustes Analysis.

## 2.3 Experiments

This section is dedicated to the description and discussion of the experiments conducted to validate the proposed method. We begin in Section 2.3.1 by describing the two datasets we used, 300W a dataset intensely used by the community and BU4DFE-S, a dataset we have specially synthesised from BU4DFE, a 3D face dataset. In Section 2.3.2 we present the experimental setup and the methods used for comparison [1]. In Section 2.3.3 we discuss the results.

The objective of these experiments are two-fold. First we want to show that the proposed method achieves state-of-the-art results on close-to-frontal faces. For this purpose we use 300W, a well known public dataset which is the de-facto standard benchmarking dataset for facial landmark localisation. We then want to show that the method outperforms other methods when applied to heavily rotated faces. For this purpose we show results on the BU4DFE-S, a dataset specially synthesised for this purpose. The reader is referred to Table 2.1 for the overall results on the two considered datasets and to Figure 2.3 for a comparative study of the robustness to rotation 2.3. Detection examples are presented in Figure 2.4. The code for the experiments is made publicly available.

### 2.3.1 Data

In order to test the proposed method we used 300W, a well known facial expression dataset. We also designed a new dataset, which we called BU4DFE-S, consisting of 2D faces synthesised from BU4DFE, a public 3D dynamic facial expression dataset.

**300W**. The 300 Faces In-the-Wild (300W) [40] database is a compilation of six re-annotated datasets (68 landmarks). Following the same approach as in [34] [38], four of the six datasets are used: AFW [41], LFPW [42], Helen [43] and iBUG [40]. The test data for LFPW and Helen, along with iBug, are used as test. The rest of data is used for training. This provides a total of 3148 and 689 train and test instances. The data is captured outside the lab and it has balanced ethnic and gender distribution. While challenging and diverse, it does not contain far-from-frontal faces and its number of samples is rather low.

**BU4DFE-S**: While annotated face datasets have become more challenging and diverse in recent years, they still provide a low number of training instances with limited variation in rotation. In order to compare the robustness of the proposed method with state-of-the-art facial landmark localisation methods, we have created BU4DFE-S, a new large 2D dataset synthesised from the publicly available BU4DFE. BU4DFE is a high resolution dynamic 3D facial dataset [44]. 101 subjects of ages between 18 to 45 years old are captured while showing facial expressions in a controlled environment. The 3D facial expressions are captured at 25 frames per second. Each sequence begins with the neutral expression, proceeds to target emotion and then back to neutral. For creating the BU4DFE-S we sample 5 frames

[1] Code and data generation script available at https://github.com/moliusimon/csdm

from each captured sequence. The sampled 3D frames are equally distributed along the sequence, portraying varying intensities of the same expression during onset, apex and offset.

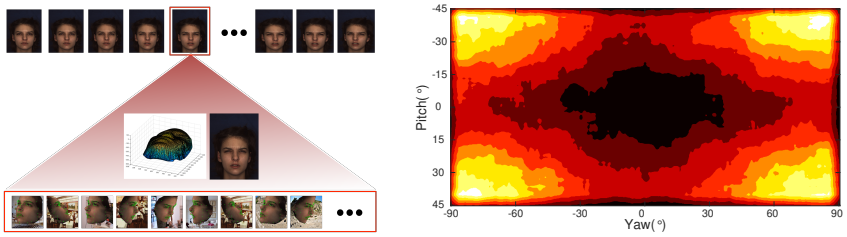
We use the extracted 3D samples, to build 25 2D projections by rotating the 3D model in pitch and yaw. The projected images are generated as follows. The BU4DFE provides the 3D point cloud of the face and an RGB image. Additionally for each of the 3D points the mapping is provided to the corresponding position on the RGB image, making it possible to map the 3D geometry to the colour texture. We first homogeneously down-sample the 3D points set by 20 and build a triangle mesh from the remaining points. The down-sampling factor was heuristically found as a trade-off between the computational cost for generating the projected images and their quality. We consider an isometric projection to associate texture patches to the mesh triangles. The mesh is then rotated with the desired angle and the triangles are again projected to the 2D plane. The new face is built by affine piece-wise warping of the initial texture patches to the newly projections of the rotated triangles by taking into account self occlusions. Inpainting is used to fill warping holes or artifacts. Finally, the images are resized to a standard size of $200 \times 200$ pixels and a background is painted on the remaining regions.

We have used the test partition of the Places Dataset, a scene recognition dataset, to build the backgrounds [45]. It contains 41000 images of size $256 \times 256$ pixels. From every image we crop two $200 \times 200$ regions, one on the top-let corner and the other on the bottom-down corner. The former is flipped. We use these images to place a different background behind each of the generated faces. In Figure 2.1(a) we provide a summarised depiction of the process. The rotation angles follow an inverse normal distribution for angles between $\pm 90°$ in yaw and $\pm 45°$ in pitch as shown in Figure 2.1(b). In this way we obtain more highly rotated faces in all directions than close-to-frontal faces. The generated data contains a total of 75000 rotated images of 100 persons. Each person appears in 750 samples with 6 different facial expressions at 5 different intensities rotated 25 times. As the BU4DFE, the subjects are from different ethnicities and follow a balanced gender distribution. The generated dataset has more instances and rotation variation than any other existing public 2D dataset. We show some examples in Figure 2.2.

Besides containing a larger number of samples (approximately 24 times more than 300W), BU4DFE-S has two more important characteristics. First, for each of the samples the pose is known which is not the case with most of the other 2D face datasets. There exist datasets containing captured faces under different angles in the lab, but the angle distribution is extremely skewed [46]. Another advantage of the BU4DFE-S is that we have total control over the pose distribution of the synthesised data. This makes possible benchmarking the robustness to pose rotation against state-of-the-art methods as shown in Figure 2.3.

### 2.3.2 Experimental settings

For the proposed method the parameter space is larger than for SDM, specially at the first cascade steps. In order to avoid over-fitting, the training data is augmented. For both the 300W and BU4DFE-S datasets the images and geometries are mirrored, doubling the number of training instances. In the case of 300W, which consists of only 3148 training images, the dataset is further augmented by providing 25 different initial geometries. These are generated by applying a random




(a) Data synthesis for BU4DFE-S. (b) Pose rotation distribution for BU4DFE-S.

Figure 2.1: BU4DFE-S contains 2D rotated faces synthesised from BU4DFE, a 3D dynamic facial expression dataset. In (a) we show how from a original sequence we sample a limited number of equally spaced frames. For each of these frames we use the provided 3D mesh and the texture to generate 25 rotated projections. The rotation angle distribution is shown in (b). We favour far-from-frontal faces with respect to close-to-frontal ones in order to make the data as challenging as possible.

Figure 2.2: BU4DFE-S dataset samples. Annotated face landmarks are shown in green.

rotation between $[-\pi/4,\ \pi/4]$, a displacement between $[-5\%,\ 5\%]$ for both width and height, and a scaling factor between $[0.9,\ 1.1]$ to the mean shape.

Regarding the number of bases $l$ and the captured feature space variance, the values have been manually chosen for each dataset. For 300W, 2 bases and 95% of variance are used, while for BU4DFE-S, 5 bases and 85% of variance are used. It is necessary to use fewer bases in 300W in order to avoid over-fitting, since the number of training instances is smaller.

We compare the proposed method with the most important facial landmark localization methods in recent years. This is done using the Normalized Mean Euclidean Error (NMEE) metric, a standard error metric in the literature [1, 33]. It corresponds to the mean euclidean distance between the detected and ground truth landmarks, normalized by the inter-ocular distance. In the case of BU4DFE-S, where large head rotations are present, the 3D inter-ocular distance is used instead. Otherwise for yaw angles close to 90° the inter-ocular distance would tend to zero, giving more weight to errors on heavily rotated faces. For comparing

| | ESR [2] | RCPR [33] | SDM [1] | ERT [35] | LBF [34] | CGPRT [38] | CFSS [3] | GSDM [24] | CSDM | CSDMa |
|---|---|---|---|---|---|---|---|---|---|---|
| 300W | 7.58 | 8.38 | 7.52 | 6.40 | 6.32 | **5.71** | 5.76 | 6.96 | 6.83 | 6.40 |
| BU4DFE-S | 9.45 | 8.61 | 9.57 | - | - | 15.81 | - | 9.01 | 8.28 | **7.62** |

Table 2.1: Our method compared with state-of-the-art methods in terms of mean landmark displacement as percentage of inter-ocular distance[2] without (CSDM) and with multiple test initializations (CSDMa).

results we considered the most important state-of-the-art methods [2], [33], [1], [24], [35], [34], [43], [3]. RCPR is able to deal with occlusions by including occlusion ground-truth of the landmarks in the learning process. As none of the considered datasets has annotated occlusions we discarded this feature during training. For the ERT [35] and LBF [34], we compare with already published results for the 300W. For a fair comparison we compare the results for the SDM and the GSDM after training with the same number of steps as the proposed method. It is important to note that GSDM is a method oriented to tracking the facial geometry, but can be easily applied to the static case by modifying the definition of the subspace used to partition the feature space. Instead of using two principal components from $\Delta\mathbf{X}^i$ and one from $\Delta\mathbf{\Phi}^i$, all principal components are taken from $\Delta\mathbf{\Phi}^i$. For the proposed approach, a 2-dimensional subspace is used in the case of 300W, and a 5-dimensional one for BU4DFE-S. Finally, two recent methods, CFSS [3] and CGPRT [38], have been considered. In their paper, the authors of CGPRT publish two results, with different number of training steps. The result we have obtained was with the larger number of steps and by initializing with the mean shape. CFSS does a constraint search of the shape in a coarse-to-fine manner in subsequent finer shape sub-spaces. Even though a parallel training on the CPU was attempted, we found training to be very slow, which made impossible obtaining results for BU4DFE-S with the available hardware resources.

### 2.3.3 Results discussion

For the 300W dataset, the trained model has been fit to the test data both using mean shape initialization and with 25 random initializations sampled using the same criteria used during training (see Section 2.3.2). The results of both approaches are shown in Table 2.1. Without multiple test initializations, the method has a NMEE lower than those achieved by ESR, RCPR, SDM and GSDM, also surpassing ERT when using multiple initializations. Yet LBF, CGPRT and CFSS still have lower errors. Thus, the proposed approach surpasses, or is close, to most state-of-the-art methods in the near frontal view conditions of the 300-w dataset.

On BU4DFE-S the proposed method outperforms all considered state-of-the-art approaches. Because it is a dataset with large head pose rotations in both pitch and yaw, this dataset better represents the strength of the proposed algorithm to better adjust to the main modes of variation of the data. This is analyzed in Figure 2.3. There, the NMEE is shown relative to the yaw rotation, for two ranges of pitch. Without using multiple test initializations, the proposed method has an accuracy similar to that of the other state-of-the-art approaches for near-frontal faces, but is much more robust to pose variations. It works specially well for both large pitch and yaw rotations. This contrasts with CGPRT, which performed specially well for the 300W dataset, but had problems with BU4DFE-S. The only method still far

[2]For the BU4DFE-S we compute the inter-ocular distance in 3D.

| | Close to frontal | | | | | | | Far from frontal | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ESR | RCPR | SDM | CGPRT | GSDM | CSDM | CSDMa | ESR | RCPR | SDM | CGPRT | GSDM | CSDM | CSDMa |
| eyes | 3.92 | **3.38** | 4.02 | 10.53 | 3.92 | 4.04 | 3.82 | 6.94 | 6.11 | 6.76 | 14.29 | 6.25 | 5.55 | **5.20** |
| eyebrows | 5.84 | **5.17** | 5.60 | 13.15 | 5.56 | 5.84 | 5.54 | 9.01 | 8.02 | 8.50 | 17.73 | 8.12 | 7.20 | **6.77** |
| nose | 6.03 | 5.59 | 5.60 | 10.30 | 5.51 | 5.58 | **5.27** | 8.26 | 7.69 | 8.58 | 13.21 | 8.00 | 7.41 | **6.99** |
| mouth | 5.46 | 4.28 | 4.47 | 10.91 | 4.27 | 4.40 | **4.27** | 8.20 | 6.70 | 8.18 | 14.52 | 6.72 | 6.17 | **5.84** |
| contour | 12.59 | 12.11 | 13.26 | 17.49 | 13.52 | 13.27 | **12.04** | 17.30 | 17.19 | 18.54 | 22.43 | 18.53 | 17.20 | **15.27** |

Table 2.2: Normalized Mean Euclidean Error (NMEE) for different landmark subsets corresponding to facial regions on BU4DFE-S. We group faces according to their pose. Close-to-frontal faces have an yaw angle between $\pm 30°$ and pitch angle between $\pm 15°$. Correspondingly far-from frontal faces have both yaw and pitch angles above $\pm 30°$ and $\pm 15°$ respectively.

from, but approaching the accuracy obtained by the proposed approach is RCPR. It can be seen in Figure 2.3 that while RCPR has the lowest NMEE for frontal faces, it one of the best approaches when dealing with large pose variations. When using multiple test initializations, a much lower average error is obtained, achieving the same accuracy for near-frontal faces as ERT. This accuracy improvement is maintained regardless of the facial pose, except for large pose rotations in both pitch and yaw, where the yaw angle is close to $90°$. For these extreme cases, the error is only slightly lower than CSDM without using multiple shape initializations.

A breakdown of the NMEE by facial regions, as shown in Table 2.2, gives a better insight on the method performance. For far from frontal head poses, the proposed approach surpasses the state-of-the-art accuracies on all facial regions, both with and without multiple test initializations. In the case of near-frontal head poses, RCPR has a higher precision for the eyes and eyebrows. CSDM is better at localizing landmarks at the nose, mouth and contour regions when using multiple shape initializations. An interesting result is the error reduction when localizing the contour landmarks with multiple shape initializations. While the other facial regions reduce the RMSE by about 5%, in the case of the contour it is reduced by over 10%, both in close to and far from frontal head poses. This is likely caused by the lack of edges and strong gradients on this region. By averaging multiple predictions, the noise is reduced, obtaining a higher accuracy.

GSDM is another method that exploits the features main modes of variation to better approximate the descent direction at different regions of the feature space. Compared to it, the proposed method obtains better results for both 300W and BU4DFE-S while also producing a more compact model. The memory required by GSDM increases quadratically with the number of considered bases, while the proposed approach does so linearly. Furthermore, each position of the subspace has a unique regressor assigned, while GSDM shares the same regression weights for a given partition of the subspace. One downside to the proposed approach is that the computational cost increases linearly with the number of bases, while for GSDM the cost remains constant.

Similarly to SDM and GSDM, the proposed method provides a closed-form solution. Compared to other state-of-the-art methods such as CFSS, CGPRT and LBF, which use stochastic processes when learning each regressor, the proposed approach ensures a consistent result on different training runs given the same data.

Multiple qualitative examples of faces from the BU4DFE-S dataset, with the landmark predictions for different methods, are shown in Figure 2.4. From these examples it can be seen that SDM, CGPRT and RCPR struggle to correctly locate inner face landmarks for heavily rotated faces. Compared to all other considered

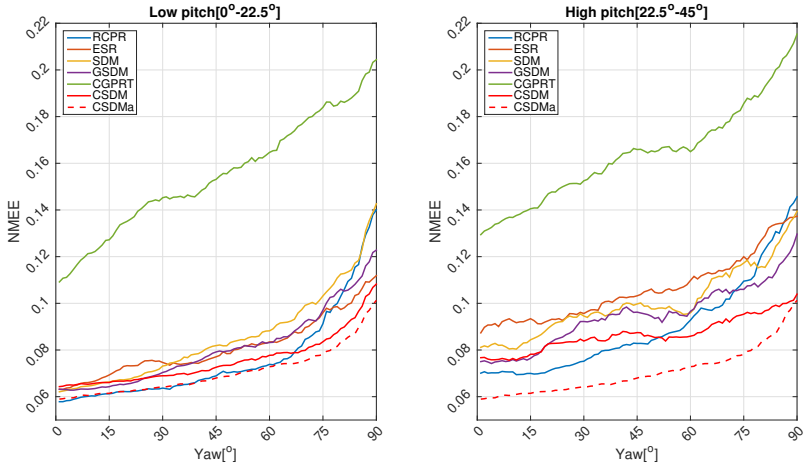



Figure 2.3: Normalized Mean Euclidean Error (NMEE) as a function of yaw on two different pitch ranges on BU4DFE-S.

methods, our proposal has a high accuracy on inner face landmarks even with highly rotated faces, followed by GSDM and ESR. The main weakness is the localization of face contour landmarks, which is noisy due to the lack of edges and little texture information on that area, resulting in a lack of smoothness in the contour line. Even with this noise, as shown in Table 2.2, the proposed approach has a much better precision for this set of landmarks. An extension to consider in the future would be regressing a parametrized shape, which should increase the accuracy for the face contours.

## 2.4 Conclusion

In this work we extended cascaded regression approaches by introducing the second order derivative over the main modes of variation of the features, presenting a closed-form solution to the face alignment problem. We showed that by doing so, the robustness to large head pose variations is greatly increased, surpassing current state-of-the-art methods. At the same time, the accuracy for near-frontal faces is comparable to state-of-the-art results. Furthermore, the learned models are smaller than those from other similar approaches.

In order to prove the effectiveness of our method on heavily rotated faces we have built a new synthetic dataset based on a well known public 3D face dataset. It contains large variations in both head pose and facial expressions, as well as a large number of training instances, making it one of the largest, most challenging datasets for facial landmark localization to date.

Several future improvements can be envisioned, like parameterizing the face to increase shape consistency especially for landmarks situated in regions with little texture and extending the method to 3D, which would make it useful for a larger number of applications.

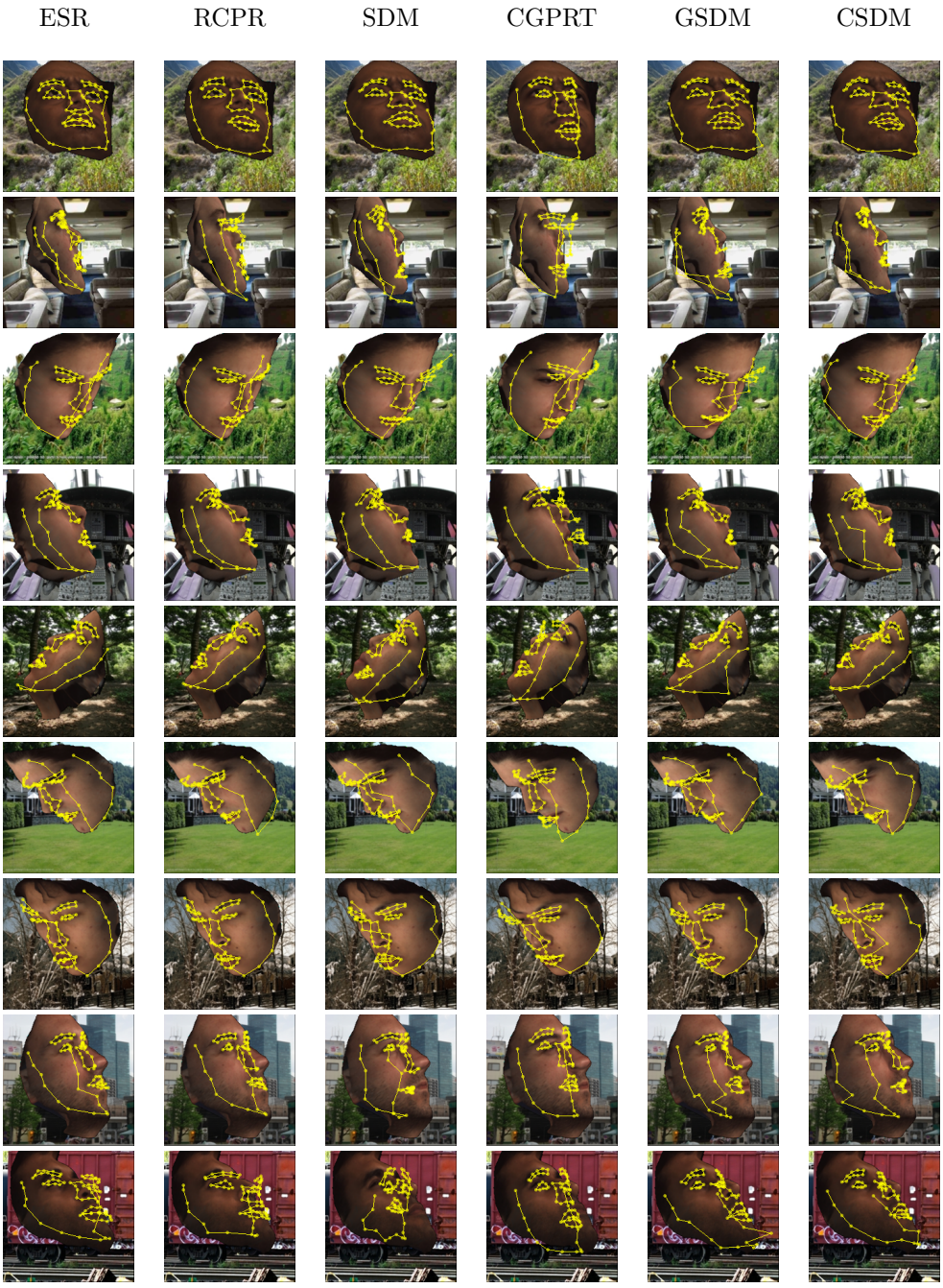


Figure 2.4: Facial landmark localisation examples for BU4DFE-S.

# Part II

# Shared-state Neural Networks

# Chapter 3

# Order relationships between features in a neural network

Before moving from a simple linear model such as cascaded regression and into models based on deep neural networks, it is important to understand how neural networks capture the different order relationships of their input data. To do so, we explore a simple neural model known as a quadratic neural network, its equivalence to polynomial networks and how these in turn directly map into regular neural networks. More specifically, we show the maximum order of the relationships captured to be equal to $2^{l-1}$, where $l$ is the total number of layers. The maximum order of the relationships captured at a given layer is thus dependent on its distance from the input.

## 3.1 Quadratic neural networks

During the past decade, various quadratic models have been proposed as a replacement for fully connected layers in a neural network [47, 48] in order to increase the expressiveness of the resulting model. One such example [47] uses a Volterra series to describe a 2nd order multiple regression model, which can be compactly represented as:

$$y_i(x) = x^T W_2 x + x^T W_1 + b \tag{3.1}$$

Here, the response $y_i(x)$ of neuron $i$ for a given input feature vector $x \in \mathcal{R}^{<m>}$ is decomposed into the summation of a quadratic term parametrized by $W_2$, a linear term parametrized by $W_1$ and a bias $b$. We choose this model as a starting point due to the ease with which different order relationships between variables are captured. One can easily see that stacking multiple such layers would result in a doubling of the maximum order of the captured relationships for each additional layer stacked.

We first rewrite the formulation in order to obtain a more compact form. To do this, we introduce a bias component to the input, such that $\tilde{x} = [x; 1]$, and use a single matrix $w$ encoding the quadratic, linear and bias components of the equation

above:

$$
\begin{aligned}
y_i(x) = \; & \tilde{x}^T W \tilde{x} \\
W = \; & \begin{bmatrix} (W_2)_{1,1} & (W_2)_{1,2} & .. & (W_1)_1 \\ (W_2)_{2,1} & (W_2)_{2,2} & .. & (W_1)_2 \\ .. & .. & .. & .. \\ (W_1)_1 & (W_1)_2 & .. & b \end{bmatrix}
\end{aligned} \tag{3.2}
$$

Please note that the resulting quadratic form is still capable of encoding linear relationships. The resulting matrix would consist on zeros everywhere except the last row and column, forming a rank 2 matrix.

## 3.2 Equivalence to polynomial networks

This kind of encoding of the quadratic weights is possible in [47] due to it being applied to convolutional kernels, reducing the amount of input features. In the general case the problem becomes intractable, due to the quadratic explosion in the number of parameters. One possible solution is to decompose the weights matrix. Please note that $W$ can be symmetric without any loss of generality, since $\tilde{x}^T W \tilde{x} = \tilde{x}^T W^T \tilde{x}$. Thus we can decompose it into a matrix of basis and their corresponding eigenvalues ($W = USU^T$) through eigendecomposition:

$$
y_i(x) = \tilde{x}^T U S U^T \tilde{x} = (\tilde{x}^T U) S (\tilde{x}^T U)^T \tag{3.3}
$$

Here $U \in \mathcal{R}^{<(m+1)\times k>}$ encodes the basis of $W$ as columns, and $S \in \mathcal{R}^{<k\times k>}$ is a diagonal matrix with the corresponding eigenvalues. One can use a subset of basis ($k \leq m+1$) to encode a rank-$k$ approximation of $W$.

The formulation in Equation 3.3 describes a single neuron taking a single input vector. Here we extend it to the multivariate case and to multiple input instances. Lets define $\overline{U} = \left[U^{(1)}, ..., U^{(n)}\right]$ as the concatenation of the basis for each neuron $U^{(i)}$. Similarly, lets define $\overline{S}$ as the diagonal matrix consisting of all $S^{(i)}$. We define a function $y(X) : \mathcal{R}^{<q\times m>} \rightarrow \mathcal{R}^{<q\times n>}$ mapping $q$ input instances to the output of $n$ quadratic neurons as:

$$
\begin{aligned}
y(X) &= \left[(X\overline{U}) \circ (X\overline{U})\right] M \\
&where \; M = \overline{S}\widetilde{M}
\end{aligned} \tag{3.4}
$$

Here $\circ$ denotes the Hadamard product. $\widetilde{M}$ is a predefined selector matrix adding the basis that define the quadratic matrix of each neuron. In the case of a standard quadratic layer, that would be a block-diagonal matrix with column vectors of ones at the diagonal.

It is easy to see from Equation 3.4 that a quadratic layer corresponds to a two-layer neural network. Given a quadratic activation function $g(x) = x^2$, Equation 3.4 can be rewritten as:

$$
y(X) = g(X\overline{U})M \tag{3.5}
$$

This equation corresponds to a two-layer neural network, where the first hidden layer $z^{(1)} = g(X\overline{U})$ is a standard layer with a quadratic activation and the second layer $z^{(2)} = z^{(1)}M$ is a layer without an activation function, performing linear

combinations of the input. Such a neural network is known as a polynomial network [49]. Some quadratic networks apply an additional activation function to the output of quadratic units. This would correspond to the activation function of the output layer in the equivalent two-layer model.

The only difference between a quadratic and polynomial network is the layout of $M$. In the case of a quadratic network, $M$ is a sparse matrix, indicating a fixed connectivity of the output layer. For polynomial networks, $M$ is a dense matrix instead. Conceptually, both quadratic and polynomial networks encode a quadratic regressor for each output neuron, with polynomial networks being able to re-use basis across regressors.

## 3.3 Equivalence to neural networks

We can go one step further and replace the input $X$ in Equation 3.4 by the equation itself, obtaining a model with two quadratic layers:

$$\begin{aligned} y(X) &= \left[\left(X^{(2)}M^{(1)}\overline{U}^{(2)}\right) \circ \left(X^{(2)}M^{(1)}\overline{U}^{(2)}\right)\right] M^{(2)} \\ X^{(2)} &= \left[(X\overline{U}^{(1)}) \circ (X\overline{U}^{(1)}),\ \mathbf{1}^{<q\times 1>}\right] \end{aligned} \tag{3.6}$$

Here $M^{(i)}$ and $\overline{U}^{(i)}$ correspond to the weighted selector matrix and quadratic regression basis for layer $i$. We defined the linear component and bias of the second layer by concatenating a column of ones to $(XU^{(1)}) \circ (XU^{(1)})$ and a new row and column to M, with all zeros except for the last entry. A point of interest in Equation 3.6 is the product $M^{(1)}U^{(2)}$. It corresponds to forming the quadratic kernels for the first layer and computing the second layer basis. Since both sets of parameters combine linearly, we can remove $M^{(1)}$ and let $\overline{U}^{(2)}$ do both jobs. This results in the following simplification:

$$\begin{aligned} y(X) &= \left[\left(X^{(2)}\overline{U}^{(2)}\right) \circ \left(X^{(2)}\overline{U}^{(2)}\right)\right] M^{(2)} \\ X^{(2)} &= \left[(X\overline{U}^{(1)}) \circ (X\overline{U}^{(1)}),\ \mathbf{1}^{<q\times 1>}\right] \end{aligned} \tag{3.7}$$

The only practical difference is the predefined combination of basis through matrices $M^i$ in quadratic networks, computing the quadratic response of each neuron. In polynomial networks basis are freely combined to form new, higher order basis, obtaining a potentially more compact model, even if the maximum order of the captured data relationships is the same.

Please note that Equation 3.7 corresponds to two layers with a quadratic activation function, followed by a linear layer without an activation. If we repeat the stacking process, we get to the more general solution where the stacking of $l$ quadratic layers are equivalent to a regular neural network with $l+1$ layers, all layers using a quadratic activation function except for the last layer, which is linear. From this observation we conclude that a regular neural network with $l$ layers is equivalent to a set of high order regressors, the equivalence being as such:

1. A regular neural network with $l$ layers is equivalent to a compact representation of $2^{l-1}$-th order regressors.

2. There are as many regressors of that order as neurons in the output layer.

3. Differently from standard $n$-th order regressors, the basis composing them are shared due to the outputs of hidden layers being shared among the neurons of the following layer.

4. In the general case, where other non-linear activation functions are used instead of quadratic activations, we have an equivalent model that, although not identical to an $n$-th order regressor, can capture the same order of variable interactions.

# Chapter 4

# Folder Recurrent Neural Networks

## 4.1 Introduction

Future video prediction is a challenging task that recently received much attention due to its capabilities for learning in an unsupervised manner, making it possible to leverage large volumes of unlabeled data for video-related tasks such as action and gesture recognition [50, 51, 52], task planning [53, 54], weather prediction [55], optical flow estimation [56] and new view synthesis [52].

One of the main problems in this task is the need of expensive models, both in terms of memory and computational power, in order to capture the variability present in video data. Another problem is the propagation of errors in recurrent models, which is tied to the inherent uncertainty of video prediction: given a series of previous frames, there are multiple feasible futures. Left unchecked, this results in blurry predictions averaging the space of possible futures. When predicting subsequent frames, the blur is propagated back into the network, accumulating errors over time.

In this work we propose a new type of recurrent auto-encoder with state sharing between encoder and decoder. We show how the exposed state in Gated Recurrent Units (GRU) can be used to create a bidirectional mapping between the input and output of each layer. To do so, the input is treated as a recurrent state, adding another set of logic gates to update it based on the output. Creating a stack of these layers allows for a bidirectional flow of information: The forward gates encode inputs and the backward ones generate predictions, obtaining a structure similar to an auto-encoder[1], but with many inherent advantages. Only the encoder or decoder is executed for input encoding or prediction, reducing memory and computational costs. Furthermore, the representation is stratified: low level information not necessary to capture higher level dynamics is not passed to the next layer. Also, it naturally provides a noisy identity mapping of the input, facilitating the initial stages of training: the input to the first dGRU holds the last encoded frame or, if preceded by convolutional layers, an over-complete representation of the same. During generation, the first untrained dGRU randomly modifies the last input, in-

[1] Code available at https://github.com/moliusimon/frnn

troducing a noise signal. While the approach does not solve the problem of blur, it prevents its magnification by mitigating the propagation of errors. Moreover, a trained network can be deconstructed to analyze the role of each layer in the final predictions, making the model more explainable. Since the states are shared, the architecture can be thought of as a recurrent auto-encoder folded in half, with encoder and decoder layers overlapping. We call our method Folded Recurrent Neural Network (fRNN).

Our main contributions are:

1. A new shared-state recurrent auto-encoder with lower memory and computational costs.
2. Mitigation of error propagation through time.
3. It naturally provides an identity function during training.
4. Model explainability and optimization through layer removal.
5. Demonstration of representation stratification.

## 4.2 Related work

Video prediction is usually approached using deep recurrent models. While initial proposals focused on predicting small patches [57, 58], it is now common to generate the whole frame based on the previous ones.

**Building Blocks**. Due to the characteristics of the problem, an auto-encoder setting has been widely used [50, 53, 11, 12, 13]: the encoder extracts information from the input and the decoder produces new frames. Usually, encoder and decoder are CNNs that tackle the spatial dimension. LSTMs are commonly used to handle the temporal dynamics and project the representations into the future. Some works compute the temporal dynamics at the deep representation bridging the encoder and decoder [53, 56, 59, 13]. Others jointly handle space and time by using Convolutional LSTMs [11, 51, 56, 60, 61] (or GRUs, as in our case), which use convolutional kernels at their gates. For instance, Lotter *et al.* [51] use a recurrent residual network with ConvLSTMs where each layer minimizes the discrepancies from previous block predictions. Common variations also include a conditional term to guide the temporal transform, such as a time differential [62] or prior knowledge of scene events, reducing the space of possible futures. Oh *et al.* [53] predict future frames on Atari games conditioning on the player action. Some works propose such action conditioned models foreseeing an application for autonomous agents learning in an unsupervised fashion [11, 60]. Finn *et al.* [11] condition their predictions for a physical system on the actions taken by a robotic arm interacting with the scene. The method was recently applied to task planning [54] and adapted to stochastic future video prediction [63].

**Bridge connections**. Introducing bridge connections (connections between equivalent layers of the encoder and decoder) is also common [11, 52, 59, 12]. This allows for a stratified representation of the input sequence, reducing the capacity needs of subsequent layers. *Video Ladder Networks* (VLN) [59] use a convolutional auto-encoder where pairs of convolutions are grouped into residual blocks. Bridge connections are added between corresponding blocks, both directly and by using a recurrent bridge layer. This topology was further extended with *Recurrent Ladder Networks* (RLN) [64], where the recurrent bridge connections were removed, and the

residual blocks replaced by recurrent layers. Using bridge connections instead of the proposed state sharing has some disadvantages: higher number of parameters and memory requirements, impossibility to skip the encoding/decoding steps (resulting in a higher computational cost) and reduced explainability due to not allowing layers to be removed after training. Finally, bridge connections do not provide an initial identity function during training. This makes it hard for the model to converge in some cases: when the background is homogeneous the model may not learn a proper initial mapping between input and output, but set the weights to zero and adjust the bias of the last layer, eliminating the gradient.

**Prediction atom**. Most of the proposed architectures for future frame prediction work at the pixel level. However, some models have been designed to predict motion and use it to project the last frame into the future. These may generate optical flow maps [52, 56] or convolutional kernels [65, 66]. Other methods propose mapping the input sequence onto predefined feature spaces, such as affine transforms [67] or human pose vectors [68]. These systems use sequences of such features to generate the next frame at the pixel level.

**Loss and GANs**. Commonly used loss functions such as L2 or MSE tend to average the space of possible futures. For this reason, some works [69, 12, 68, 61] propose using Generative Adversarial Networks (GAN) [70] to aid in the generation of realistic looking frames and coherent sequences. Mathieu *et al.* [69] use a plain multi-scale CNN in an adversarial setting and propose the Gradient Difference Loss to sharpen the predictions.

**Disentangled Motion/Content**. Some authors encode content and motion separately. Villegas *et al.* [12] use an auto-encoder architecture with a two-stream encoder: for motion, a CNN + LSTM encodes difference images; for appearance, a CNN encodes the last input frame. The decoder receives a concatenation of both and uses multi-scale residual connections. In a similar fashion, Denton *et al.* [13] use two separate encoders and an adversarial setting to obtain a disentangled representation of content and motion. Alternatively, some works predict motion and content in parallel to benefit from the combined strengths of both tasks. While Sedaghat *et al.* [71] propose using a single representation with a dual objective (optical flow and future frame prediction), Liang *et al.* [61] use a dual GAN setting and use predicted motion to refine the future frame prediction.

**Feedback Predictions**. Finally, most recurrent models are based on the use of feedback predictions: they input previous predictions to generate subsequent frames. If not handled properly, this may cause errors to accumulate and magnify over time. Our model mitigates this by enabling encoder and decoder to be executed any number of times independently. This is similar to the proposal by Srivastava *et al.* [50], which uses a recurrent auto-encoder approach where an input sequence is encoded and its state copied into the decoder. The decoder is then applied to generate a given number of frames. However, it is limited to a single recurrent layer at each part.

Here, stochastic video prediction is not considered. Such models learn and sample from a space of possible futures to generate the following frames. This reduces prediction blur by preventing the averaging of possible futures. fRNN could be extended to perform stochastic sampling by adding an inference model similar to that in [63] during training. Samples drawn from the predicted distribution would be placed into the deepest state of the dGRU stack. However, this would make it difficult to analyse the contribution of dGRU layers to the mitigation and recovery

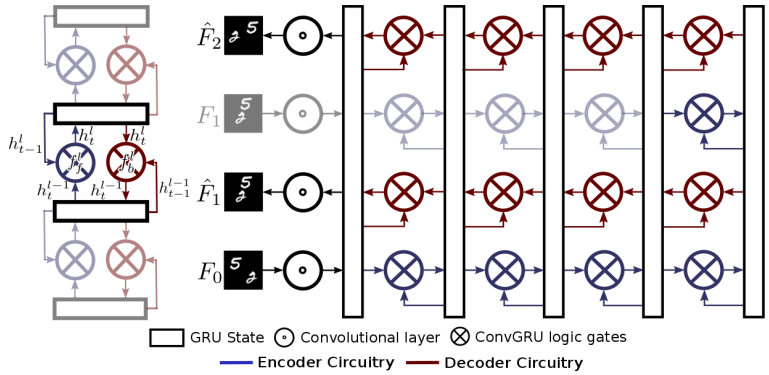



Figure 4.1: **Left:** Scheme of a dGRU. Shadowed areas illustrate additional dGRU layers. **Right:** fRNN topology. State cells are shared between encoder and decoder, creating a bidirectional state mapping. Shadowed areas represent unnecessary circuitry: re-encoding of the predictions is avoided due to the decoder updating all the states.

from blur propagation.

## 4.3 Proposed method

We propose an architecture based on recurrent convolutional auto-encoders to deal with the network capacity and error propagation problems for future video prediction. It is built by stacking multiple double-mapping GRU layers, which allow for a bidirectional flow of information between input and output: they consider the input as a recurrent state and update it using an extra set of gates. These are then stacked, forming an encoder and decoder using, respectively, the forward and backward gates (Figure 4.1). We call this architecture Folded Recurrent Neural Network (fRNN). Because of the state sharing between encoder and decoder, the topology allows for: stratification of the representation, lower memory and computational requirements compared to regular recurrent auto-encoders, mitigated propagation of errors, and increased explainability through layer removal.

### 4.3.1 Double-mapping Gated Recurrent Units

GRUs have their state fully exposed as output. This allows us to define a bidirectional mapping between input and output by replicating the logic gates of the GRU layer. To do so, we consider the input as a state. Lets define the output of a GRU at layer $l$ and time step $t$ as $h_t^l = f_f^l(h_t^{l-1}, h_{t-1}^l)$ given an input $h_t^{l-1}$ and its state at the previous time step $h_{t-1}^l$. A second set of weights can be used to define an inverse mapping $h_t^{l-1} = f_b^l(h_t^l, h_{t-1}^{l-1})$ using the output of the forward function at the current time step to update its input, which is treated as the hidden state of the inverse function. This is illustrated in Figure 4.1. We will refer to this bidirectional mapping as a double-mapping GRU (dGRU).

### 4.3.2 Folded Recurrent Neural Network

By stacking multiple dGRUs, a recurrent auto-encoder is obtained. Given $n$ dGRUs, the encoder is defined by the set of forward functions $E = \{f_f^1,\ ...,\ f_f^n\}$ and the decoder by the set of backward functions $D = \{f_b^n,\ ...,\ f_b^1\}$. This is illustrated in Figure 4.1, and is equivalent to a recurrent auto-encoder, but with shared states, having 3 main advantages:

1. It is not necessary to feed the predictions back into the network in order to generate the following predictions. Because of state sharing, the decoder already updates all the states except for the bridge state between encoder and decoder, which is updated by applying the last layer of the encoder before decoding. The shadowed area in Figure 4.1 shows the section of the computational graph that is not required when performing multiple sequential predictions. For the same reason, when considering multiple sequential elements before prediction, only the encoder is required.
2. Since the network updates its states from the higher level representations to the lowest ones during prediction, errors introduced at a given layer are not propagated into deeper layers, leaving higher-level dynamics unaffected.
3. The model implicitly provides a noisy identity function during training: the input state of the first dGRU layer is either the input image itself, when preceded by convolutional layers, or an over-complete representation of the same. A noise signal is then introduced to the representation by the backward function of the untrained first dGRU layer. This is exemplified in Figure 4.7, when all dGRU layers are removed. As shown in Section 4.4.3, this helps the model to converge on MMNIST: when the same background is shared across instances, it prevents the model from killing the gradients by adjusting the biases to match the background and setting the weights to zero.

This approach shares some similarities with VLN [59] and RLN [64]. As with them, part of the information can be passed directly between corresponding layers of the encoder and decoder, not having to encode a full representation of the input into the deepest layer. However, our model implicitly passes the information through the shared recurrent states, making bridge connections unnecessary. When compared against an equivalent recurrent auto-encoder with bridge connections, this results in lower computational and memory costs. More specifically, the number of weights in a pair of forward and backward functions is equal to $3(\overline{h^{l-1}}^2 + \overline{h^l}^2 + 2\overline{h^{l-1}}\ \overline{h^l})$ in the case of dGRU, where $\overline{h^l}$ corresponds to the state size of layer $l$. When using bridge connections, that value is increased to $3(\overline{h^{l-1}}^2 + \overline{h^l}^2 + 4\overline{h^{l-1}}\ \overline{h^l})$. This corresponds to an overhead of 44% in the number of parameters when one state has double the size of the other, and of 50% when they have the same size. Furthermore, both the encoder and decoder must be applied at each time step. Thus, memory usage is doubled and computational cost is increased by a factor of between 2.88 and 3.

### 4.3.3 Training Folded RNNs

In a regular recurrent auto-encoder, a ground truth frame is introduced at each time step by applying both encoder and decoder. The output is used as a supervision point, comparing it to the next ground truth frame. This implies all predictions are at a single time step from the last ground truth input.

| | Conv 1 | Conv 2 | Pool 1 | dGRU 1 | dGRU 2 | Pool 2 | dGRU 3 | dGRU 4 | Pool 3 | dGRU 5 | dGRU 6 | Pool 4 | dGRU 7 | dGRU 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Num. Units | 32 | 64 | - | 128 | 128 | - | 256 | 256 | - | 512 | 512 | - | 256 | 256 |
| Kernel size | $5 \times 5$ | $5 \times 5$ | $2 \times 2$ | $5 \times 5$ | $5 \times 5$ | $2 \times 2$ | $5 \times 5$ | $5 \times 5$ | $2 \times 2$ | $3 \times 3$ | $3 \times 3$ | $2 \times 2$ | $3 \times 3$ | $3 \times 3$ |
| Stride | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 |
| Activation | tanh | tanh | - | sigmoid & tanh | | - | sigmoid & tanh | | - | sigmoid & tanh | | - | sigmoid & tanh | |

Table 4.1: Parameters of the topology used for the experiments. The decoder applies the same topology in reverse, using nearest neighbors interpolation and transposed convolutions to revert the pooling and convolutional layers.

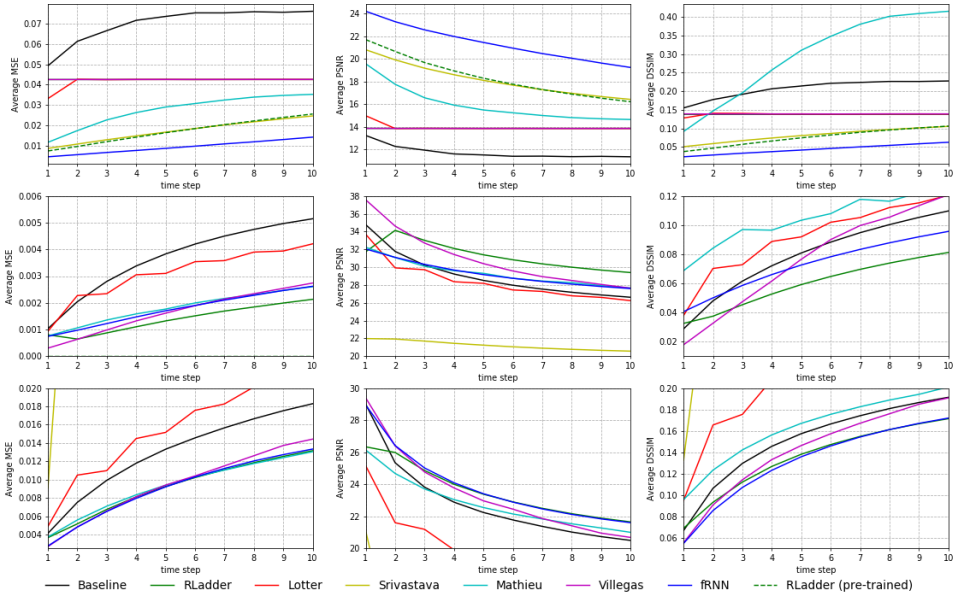



Figure 4.2: Quantitative results on the considered datasets in terms of the number of time steps since the last input frame. From top to bottom: MMNIST, KTH, and UCF101. From left to right: MSE, PSNR, and DSSIM. For MMNIST, RLadder is pre-trained to learn an initial identity mapping, allowing it to converge.

We propose a training approach for fRNNs that exploits their ability to skip the encoder or decoder at a given time step. First $g$ ground truth frames are passed to the encoder. The decoder is then applied $p$ times, producing $p$ predictions. This uses up only half the memory: either encoder or decoder is applied at each step, never both. This has the same advantage as the approach by Srivastava [50], where recurrently applying the decoder without further ground truth inputs encourages the network to learn video dynamics. This also prevents the network from learning an identity model, i.e. copying the last input to the output.

## 4.4 Experiments

In this section, we first discuss the data, evaluation protocol, and methods. We then provide quantitative and qualitative evaluations. We finish with a brief analysis on the stratification of the sequence representation among dGRU layers.

| | MMNIST | | | KTH | | | UCF101 | | |
|---|---|---|---|---|---|---|---|---|---|
| | MSE | PSNR | DSSIM | MSE | PSNR | DSSIM | MSE | PSNR | DSSIM |
| Baseline | 0.06989 | 11.745 | 0.20718 | 0.00366 | 29.071 | 0.07900 | 0.01294 | 22.859 | 0.15043 |
| RLadder | 0.04254 | 13.857 | 0.13788 | **0.00139** | **31.268** | **0.05945** | 0.00918 | 23.558 | 0.13395 |
| Lotter [51] | 0.04161 | 13.968 | 0.13825 | 0.00309 | 28.424 | 0.09170 | 0.01550 | 19.869 | 0.21389 |
| Srivastava [50] | 0.01737 | 18.183 | 0.08164 | 0.00995 | 21.220 | 0.19860 | 0.14866 | 10.021 | 0.42555 |
| Mathieu [69] | 0.02748 | 15.969 | 0.29565 | 0.00180 | 29.341 | 0.10410 | 0.00926 | 22.781 | 0.16262 |
| Villegas [12] | 0.04254 | 13.857 | 0.13896 | 0.00165 | 30.946 | 0.07657 | 0.00940 | 23.457 | 0.14150 |
| fRNN | **0.00947** | **21.386** | **0.04376** | 0.00175 | 29.299 | 0.07251 | **0.00908** | **23.872** | **0.13055** |

Table 4.2: Average results over 10 time steps.

### 4.4.1 Data and evaluation protocol

Three datasets of different complexity are considered: Moving MNIST (MMNIST) [50], KTH [72], and UCF101 [73]. MMNIST consists of $64 \times 64$ grayscale sequences of length 20 displaying pairs of digits moving around the image. We generated a million training samples by randomly sampling pairs of digits and trajectories. The test set is fixed and contains 10000 sequences. KTH consists of 600 videos of 15-20 seconds with 25 subjects performing 6 actions in 4 different settings. The videos are grayscale, at a resolution of $120 \times 160$ pixels and 25 fps. The dataset has been split into subjects 1 to 16 for training, and 17 to 25 for testing, resulting in 383 and 216 sequences, respectively. Frame size is reduced to $64 \times 80$ by removing 5 pixels from the left and right borders and using bilinear interpolation. UCF101 displays 101 actions, such as playing instruments, weight lifting or sports. It is the most challenging dataset considered, with a high intra-class variability. It contains 9950 training and 3361 test sequences. These are RGB at a resolution of $320 \times 240$ pixels and 25 fps. The frame size is reduced to $64 \times 85$ and the frame rate halved to magnify frame differences.

All methods are tested using 10 input frames to generate the following 10 frames. We use 3 common metrics for video prediction analysis: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Dissimilarity (DSSIM). MSE and PSNR are objective measurements of reconstruction quality. DSSIM is a measure of the perceived quality. For DSSIM we use a Gaussian sliding window of size $11 \times 11$ and $\sigma = 1.5$.

### 4.4.2 Methods

The proposed method was trained using RMSProp with a learning rate of 0.0001 and a batch size of 12, sampling a random sub-sequence at each epoch. Weights were orthogonally initialized and biases set to 0. For testing, all sub-sequences of length 20 were considered. Our network topology consists of two convolutional layers followed by 8 convolutional dGRU layers, applying a $2 \times 2$ max pooling every 2 layers. Topology details are shown in Table 4.1. The convolutional and max pooling layers are reversed by using transposed convolutions and nearest neighbors interpolation, respectively. We train with an L1 loss.

For evaluation, we include a stub baseline model predicting the last input frame, and a second baseline (RLadder) to evaluate the advantages of using state sharing. RLadder has the same topology as the fRNN model, but uses bridge connections instead of state sharing. Note that to keep the same state size on ConvGRU layers, using bridge connections doubles the memory size and almost triples the computational cost (Sec.4.3.2). This is similar to how RLN [64] works, but using regular

ConvGRU layers in the decoder. We also compare against Srivastava [50] and Mathieu [69]. The former handles only the temporal dimension with LSTMs, while the latter uses a 3D-CNN, not providing memory management mechanisms. Next, we compare against Villegas [12], which, contrary to our proposal, uses feedback predictions. Finally, we compare against Lotter *et al.* [51] which is based on residual error reduction. All of them were adapted to train using 10 frames as input and predicting the next 10, using the topologies and parameters defined by the authors.

### 4.4.3 Quantitative analysis

The first row of Figure 4.2 displays the results for the MMNIST dataset for the considered methods, with their numeric values shown in Table 4.3. Mean scores are shown in Table 4.2. fRNN performs best on all time steps and metrics, followed by Srivastava *et al.* [50]. These two are the only methods to provide valid predictions on this dataset: Mathieu *et al.* [69] progressively blurs the digits, while the other methods predict a black frame. This is caused by a loss of gradient during the first training stages. On more complex datasets the methods start by learning an identity function, then refining the results. This is possible since in many sequences most of the frame remains unchanged. In the case of MMNIST, where the background is homogeneous, it is easier for the models to set the weights of the output layer to zero and set the biases to match the background colour. This truncates the gradient and prevents further learning. Srivastava *et al.* [50] use an auxiliary decoder to reconstruct the input frames, forcing the model to learn an identity function. This, as discussed at the end of Section 4.3.2, is implicitly handled in our method, giving an initial solution to improve on and preventing the models from learning a black image. In order to verify this effect, we pre-trained RLadder on the KTH dataset and then fine-tuned it on the MMNIST dataset. While KTH has different dynamics, the initial step to solve the problem remains: providing an identity function. As shown in Figure 4.2 (dashed lines), this results in the model converging, with an accuracy comparable to Srivastava *et al. [50]* for the 3 evaluation metrics.

On the KTH dataset, Table 4.2 shows the best approach is our RLadder baseline followed by fRNN and Villegas *et al.* [12], both having similar results, but with Villegas *et al.* having slightly lower MSE and higher PSNR, and fRNN a lower DSSIM. While both approaches obtain comparable average results, the error increases faster over time in the case of Villegas *et al.* (second row in Figure 4.2). Mathieu obtains good scores for MSE and PSNR, but has a much worse DSSIM. Step-by-step quantitative results are shown in Table 4.3.

For the UCF101 dataset, as seen in Table 4.2, our fRNN approach is the best performing for all 3 metrics. At third row of Figure 4.2 one can see that Villegas *et al.* start out with results similar to fRNN on the first frame, but as in the case of KTH and MMNIST, the predictions degrade faster. Two methods display low performance in most cases. Lotter *et al.* work well for the first predicted frame in the case of KTH and UCF101, but the error rapidly increases on the following predictions. This is due to a magnification of prediction artifacts, making the method unable to predict multiple frames without supervision. In the case of Srivastava *et al.* the problem is about capacity: it uses fully connected LSTM layers, making the number of parameters explode quickly with the state cell size. This severely limits the representation capacity for complex datasets such as KTH and UCF101. Step-by-step quantitative results are shown in Table 4.3.

| | t=1 | t=2 | t=3 | t=4 | t=5 | t=6 | t=7 | t=8 | t=9 | t=10 | avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean Squared Error | | | | | | | | | | | |
| Baseline | 0.04923 | 0.06121 | 0.06642 | 0.07147 | 0.07338 | 0.07512 | 0.07511 | 0.07567 | 0.07542 | 0.07590 | 0.06989 |
| RLadder | 0.04251 | 0.04252 | 0.04252 | 0.04254 | 0.04255 | 0.04253 | 0.04254 | 0.04255 | 0.04255 | 0.04254 | 0.04254 |
| prednet | 0.03320 | 0.04264 | 0.04247 | 0.04254 | 0.04255 | 0.04253 | 0.04254 | 0.04255 | 0.04255 | 0.04255 | 0.04161 |
| Srivastava | 0.00885 | 0.01097 | 0.01308 | 0.01496 | 0.01680 | 0.01856 | 0.02035 | 0.02185 | 0.02335 | 0.02473 | 0.01737 |
| Mathieu | 0.08091 | 0.12012 | 0.15610 | 0.18058 | 0.19909 | 0.21074 | 0.22253 | 0.23232 | 0.23798 | 0.24156 | 0.18818 |
| Villegas | 0.04251 | 0.04252 | 0.04252 | 0.04254 | 0.04255 | 0.04253 | 0.04254 | 0.04255 | 0.04255 | 0.04255 | 0.04254 |
| fRNN | 0.00475 | 0.00578 | 0.00686 | 0.00784 | 0.00887 | 0.00994 | 0.01105 | 0.01207 | 0.01319 | 0.01435 | **0.00947** |
| PSNR | | | | | | | | | | | |
| Baseline | 13.233 | 12.266 | 11.937 | 11.601 | 11.513 | 11.396 | 11.407 | 11.362 | 11.388 | 11.350 | 11.745 |
| RLadder | 13.860 | 13.859 | 13.860 | 13.858 | 13.856 | 13.858 | 13.856 | 13.855 | 13.855 | 13.856 | 13.857 |
| prednet | 14.977 | 13.849 | 13.864 | 13.857 | 13.856 | 13.858 | 13.856 | 13.855 | 13.855 | 13.856 | 13.968 |
| Srivastava | 20.809 | 19.916 | 19.177 | 18.601 | 18.103 | 17.681 | 17.276 | 16.960 | 16.671 | 16.421 | 18.183 |
| Mathieu | 14.041 | 12.742 | 11.887 | 11.422 | 11.113 | 10.935 | 10.764 | 10.629 | 10.555 | 10.508 | 11.460 |
| Villegas | 13.860 | 13.859 | 13.860 | 13.858 | 13.856 | 13.858 | 13.856 | 13.855 | 13.855 | 13.856 | 13.857 |
| fRNN | 24.208 | 23.287 | 22.566 | 21.983 | 21.455 | 20.949 | 20.471 | 20.060 | 19.634 | 19.242 | **21.386** |
| DSSIM | | | | | | | | | | | |
| Baseline | 0.15520 | 0.17771 | 0.19192 | 0.20677 | 0.21422 | 0.22155 | 0.22383 | 0.22647 | 0.22637 | 0.22770 | 0.20718 |
| RLadder | 0.13797 | 0.13776 | 0.13783 | 0.13785 | 0.13780 | 0.13777 | 0.13789 | 0.13799 | 0.13802 | 0.13791 | 0.13788 |
| prednet | 0.12801 | 0.14085 | 0.14044 | 0.13906 | 0.13913 | 0.13886 | 0.13899 | 0.13908 | 0.13910 | 0.13899 | 0.13825 |
| Srivastava | 0.05095 | 0.05916 | 0.06735 | 0.07426 | 0.08072 | 0.08661 | 0.09239 | 0.09707 | 0.10150 | 0.10544 | 0.08164 |
| Mathieu | 0.14630 | 0.23426 | 0.31246 | 0.41025 | 0.49469 | 0.55449 | 0.60664 | 0.64062 | 0.65243 | 0.66212 | 0.47143 |
| Villegas | 0.13905 | 0.13885 | 0.13891 | 0.13894 | 0.13889 | 0.13886 | 0.13898 | 0.13908 | 0.13910 | 0.13899 | 0.13896 |
| fRNN | 0.02375 | 0.02854 | 0.03336 | 0.03762 | 0.04180 | 0.04612 | 0.05047 | 0.05444 | 0.05871 | 0.06275 | **0.04376** |

Table 4.3: Quantitative results on the Moving MNIST dataset, relative to the number of time steps since the last input frame and on average over all time steps.

| | t=1 | t=2 | t=3 | t=4 | t=5 | t=6 | t=7 | t=8 | t=9 | t=10 | avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean Squared Error | | | | | | | | | | | |
| Baseline | 0.00103 | 0.00204 | 0.00280 | 0.00338 | 0.00383 | 0.00420 | 0.00450 | 0.00475 | 0.00497 | 0.00515 | 0.00366 |
| RLadder | 0.00080 | 0.00064 | 0.00087 | 0.00110 | 0.00132 | 0.00151 | 0.00169 | 0.00184 | 0.00199 | 0.00213 | **0.00139** |
| prednet | 0.00093 | 0.00227 | 0.00234 | 0.00305 | 0.00310 | 0.00354 | 0.00358 | 0.00390 | 0.00394 | 0.00421 | 0.00309 |
| Srivastava | 0.00839 | 0.00852 | 0.00893 | 0.00940 | 0.00983 | 0.01024 | 0.01061 | 0.01093 | 0.01121 | 0.01145 | 0.00995 |
| Mathieu | 0.00518 | 0.00725 | 0.00927 | 0.01083 | 0.01208 | 0.01365 | 0.01480 | 0.01575 | 0.01687 | 0.01781 | 0.01235 |
| Villegas | 0.00030 | 0.00063 | 0.00098 | 0.00132 | 0.00161 | 0.00189 | 0.00214 | 0.00234 | 0.00254 | 0.00274 | 0.00165 |
| fRNN | 0.00074 | 0.00097 | 0.00122 | 0.00147 | 0.00170 | 0.00190 | 0.00210 | 0.00228 | 0.00246 | 0.00262 | 0.00175 |
| PSNR | | | | | | | | | | | |
| Baseline | 34.780 | 31.774 | 30.223 | 29.233 | 28.515 | 27.973 | 27.535 | 27.180 | 26.878 | 26.619 | 29.071 |
| RLadder | 31.737 | 34.142 | 33.018 | 32.109 | 31.393 | 30.834 | 30.375 | 30.000 | 29.677 | 29.397 | **31.268** |
| prednet | 33.701 | 29.922 | 29.721 | 28.365 | 28.188 | 27.439 | 27.275 | 26.782 | 26.607 | 26.238 | 28.42384 |
| Srivastava | 21.974 | 21.922 | 21.691 | 21.439 | 21.234 | 21.048 | 20.892 | 20.762 | 20.653 | 20.559 | 21.220 |
| Mathieu | 23.139 | 22.307 | 21.603 | 21.245 | 21.036 | 20.634 | 20.412 | 20.287 | 20.051 | 19.852 | 21.056 |
| Villegas | 37.575 | 34.621 | 32.709 | 31.430 | 30.401 | 29.575 | 28.940 | 28.509 | 28.061 | 27.640 | 30.946 |
| fRNN | 32.044 | 31.106 | 30.320 | 29.683 | 29.165 | 28.749 | 28.400 | 28.097 | 27.829 | 27.596 | 29.299 |
| DSSIM | | | | | | | | | | | |
| Baseline | 0.02873 | 0.04799 | 0.06156 | 0.07211 | 0.08091 | 0.08836 | 0.09483 | 0.10040 | 0.10536 | 0.10978 | 0.07900 |
| RLadder | 0.03249 | 0.03745 | 0.04533 | 0.05268 | 0.05916 | 0.06473 | 0.06965 | 0.07395 | 0.07780 | 0.08125 | **0.05945** |
| prednet | 0.03792 | 0.07026 | 0.07281 | 0.08883 | 0.09194 | 0.10199 | 0.10523 | 0.11213 | 0.11533 | 0.12057 | 0.09170 |
| Srivastava | 0.18878 | 0.18911 | 0.19203 | 0.19530 | 0.19809 | 0.20076 | 0.20307 | 0.20497 | 0.20655 | 0.20788 | 0.19868 |
| Mathieu | 0.10955 | 0.13427 | 0.15465 | 0.15397 | 0.16485 | 0.17209 | 0.18781 | 0.18569 | 0.19499 | 0.20200 | 0.16599 |
| Villegas | 0.01778 | 0.03261 | 0.04741 | 0.06162 | 0.07656 | 0.09009 | 0.09973 | 0.10550 | 0.11346 | 0.12094 | 0.07657 |
| fRNN | 0.04057 | 0.05004 | 0.05858 | 0.06605 | 0.07262 | 0.07830 | 0.08335 | 0.08787 | 0.09200 | 0.09571 | 0.07251 |

Table 4.4: Quantitative results on the KTH dataset, relative to the number of time steps since the last input frame and on average over all time steps.

Overall, for the considered methods, fRNN is the best performing on MMINST and UCF101, the latter being the most complex of the 3 datasets. We achieved these results with a simple topology: apart from the proposed dGRU layers, we use conventional max pooling with an L1 loss. There are no normalization or regularization mechanisms, specialized activation functions, complex topologies or image transform operators. In the case of MMNIST, fRNN shows the ability to find a valid initial representation and converges to good predictions where most

| | t=1 | t=2 | t=3 | t=4 | t=5 | t=6 | t=7 | t=8 | t=9 | t=10 | avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean Squared Error | | | | | | | | | | | |
| Baseline | 0.00412 | 0.00751 | 0.00992 | 0.01179 | 0.01332 | 0.01456 | 0.01566 | 0.01665 | 0.01753 | 0.01830 | 0.01294 |
| RLadder | 0.00365 | 0.00516 | 0.00674 | 0.00811 | 0.00925 | 0.01022 | 0.01108 | 0.01185 | 0.01254 | 0.01316 | 0.00918 |
| prednet | 0.00488 | 0.01049 | 0.01098 | 0.01449 | 0.01516 | 0.01757 | 0.01827 | 0.02013 | 0.02075 | 0.02226 | 0.01550 |
| Srivastava | 0.00908 | 0.05399 | 0.11943 | 0.16735 | 0.18014 | 0.17885 | 0.18194 | 0.19184 | 0.19989 | 0.20404 | 0.14866 |
| Mathieu | 0.02550 | 0.03821 | 0.04854 | 0.05707 | 0.06425 | 0.07043 | 0.07560 | 0.08049 | 0.08490 | 0.08933 | 0.06343 |
| Villegas | 0.00268 | 0.00482 | 0.00655 | 0.00812 | 0.00940 | 0.01040 | 0.01150 | 0.01261 | 0.01373 | 0.01443 | 0.00942 |
| fRNN | 0.00274 | 0.00481 | 0.00652 | 0.00795 | 0.00920 | 0.01029 | 0.01122 | 0.01204 | 0.01273 | 0.01334 | **0.00908** |
| PSNR | | | | | | | | | | | |
| Baseline | 28.993 | 25.335 | 23.812 | 22.874 | 22.230 | 21.765 | 21.362 | 21.007 | 20.722 | 20.486 | 22.859 |
| RLadder | 26.332 | 25.979 | 24.851 | 23.992 | 23.373 | 22.893 | 22.490 | 22.154 | 21.876 | 21.641 | 23.558 |
| prednet | 25.145 | 21.592 | 21.178 | 19.893 | 19.565 | 18.877 | 18.622 | 18.173 | 17.986 | 17.665 | 19.869 |
| Srivastava | 21.077 | 13.125 | 9.876 | 8.515 | 8.212 | 8.212 | 8.094 | 7.849 | 7.670 | 7.577 | 10.021 |
| Mathieu | 18.753 | 17.702 | 17.007 | 16.531 | 16.175 | 15.887 | 15.662 | 15.446 | 15.258 | 15.063 | 16.348 |
| Villegas | 29.389 | 26.389 | 24.759 | 23.765 | 22.959 | 22.440 | 21.854 | 21.401 | 20.940 | 20.671 | 23.457 |
| fRNN | 28.942 | 26.411 | 25.011 | 24.086 | 23.402 | 22.878 | 22.453 | 22.111 | 21.830 | 21.593 | **23.872** |
| DSSIM | | | | | | | | | | | |
| Baseline | 0.06650 | 0.10611 | 0.12946 | 0.14549 | 0.15735 | 0.16643 | 0.17415 | 0.18086 | 0.18654 | 0.19142 | 0.15043 |
| RLadder | 0.06874 | 0.09301 | 0.11197 | 0.12665 | 0.13804 | 0.14714 | 0.15475 | 0.16118 | 0.16664 | 0.17136 | 0.13395 |
| prednet | 0.09469 | 0.16543 | 0.17541 | 0.20803 | 0.21797 | 0.23701 | 0.24533 | 0.25795 | 0.26423 | 0.27286 | 0.21389 |
| Srivastava | 0.13221 | 0.34915 | 0.41874 | 0.46370 | 0.47410 | 0.47660 | 0.47985 | 0.48424 | 0.48746 | 0.48940 | 0.42555 |
| Mathieu | 0.15230 | 0.19642 | 0.22664 | 0.24905 | 0.26633 | 0.27976 | 0.29111 | 0.30113 | 0.30967 | 0.32065 | 0.25930 |
| Villegas | 0.05502 | 0.09051 | 0.11414 | 0.13304 | 0.14620 | 0.15725 | 0.16712 | 0.17599 | 0.18492 | 0.19083 | 0.14150 |
| fRNN | 0.05446 | 0.08526 | 0.10694 | 0.12300 | 0.13569 | 0.14580 | 0.15421 | 0.16115 | 0.16700 | 0.17195 | **0.13055** |

Table 4.5: Quantitative results on the UCF-101 dataset, relative to the number of time steps since the last input frame and on average over all time steps.

other methods fail. In the case of KTH, fRNN has an overall accuracy comparable to that of Villegas *et al.*, being more stable over time. It is only surpassed by the proposed RLadder baseline, a method equivalent to fRNN but with 2 and 3 times more memory and computational requirements.

### 4.4.4 Qualitative analysis

In this section we evaluate our approach qualitatively, observing how the model behaves in terms of predicting movement dynamics, reconstructing occluded regions and specially on the propagation of blur. We then compare predictions from our approach against those of other state-of-the-art methods.

We evaluate our approach on some samples from the three considered datasets. Figure 4.3 shows the last 5 input frames from some MMNIST sequences along with the next 10 ground truth frames and their corresponding fRNN predictions. As shown, the digits maintain their sharpness across the sequence of predictions. Also, the bounces at the edges of the image are correctly predicted and the digits do not distort or deform when crossing each other. This shows the network internally encodes the appearance of each digit, facilitating their reconstruction after sharing the same region in the image plane.

Qualitative examples of fRNN predictions on the KTH dataset are shown in Figure 4.4. It shows three actions: hand waving, walking, and boxing. The blur stops increasing after the first three predictions, generating plausible motions for the corresponding actions while background artifacts are not introduced. Although the movement patterns for each type of action have a wide range of variability on its trajectory, dGRU gives relatively sharp predictions for the limbs. The first and third examples also show the ability of the model to recover from blur. The blur slightly increases for the arms while the action is performed, but decreases again as these reach the final position.

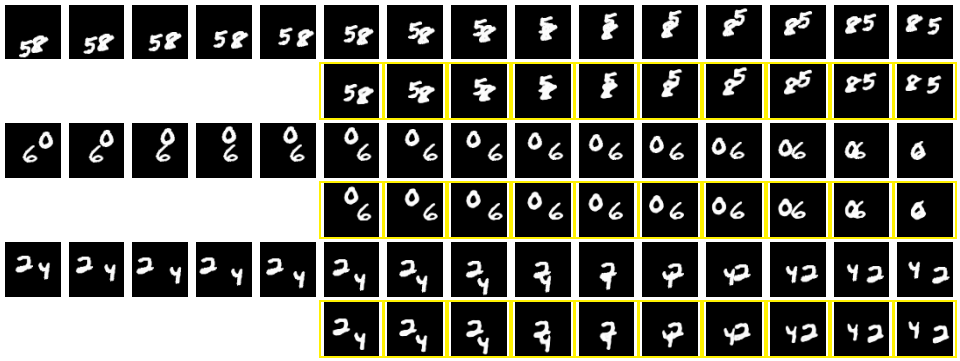


Figure 4.3: fRNN prediction examples on MMNIST. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

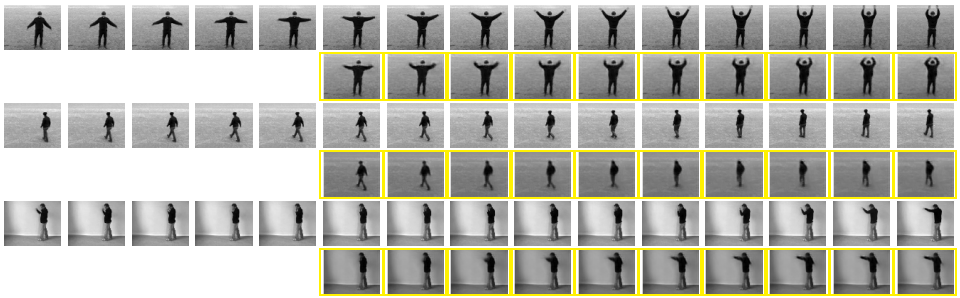


Figure 4.4: fRNN prediction examples on KTH. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

Figure 4.5 shows fRNN predictions on the UCF101 dataset. These correspond to two different physical exercises and a girl playing the piano. Common to all predictions, the static parts do not lose sharpness over time, and the background is properly reconstructed after an occlusion. The network correctly predicts actions with low variability, as shown in rows 1-2, where a repetitive movement is performed, and in the last row, where the girl recovers a correct body posture. Blur is introduced to these dynamic regions due to uncertainty, averaging the possible futures. The first row also shows an interesting behavior: while the woman is standing up the upper body becomes blurry, but the frames sharpen again as the woman finishes her motion. Since the model does not propagate errors to deeper layers nor makes use of previous predictions for the following ones, the introduction of blur does not imply it will be propagated. In this example, while the middle motion could have multiple predictions depending on the movement pace and the inclination of the body, the final body pose has lower uncertainty.

In Figure 4.6 we compare predictions from the proposed approach against the RLadder baseline and other state-of-the-art methods. For the MMNIST dataset we do not consider Villegas *et al.* and Lotter *et al.* since these methods fail to successfully converge and they predict a sequence of black frames. From the rest of approaches, fRNN obtains the best predictions, with little blur or distortion. The RLadder baseline is the second best approach. It does not introduce blur, but heavily deforms the digits after they cross. Srivastava *et al.* and Mathieu *et al.*

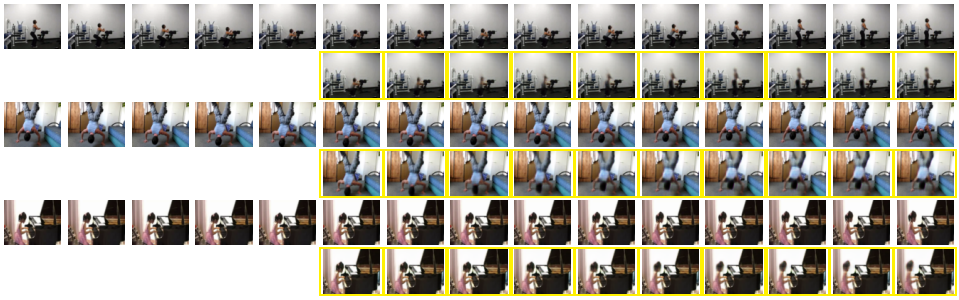


Figure 4.5: fRNN prediction examples on UCF101. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

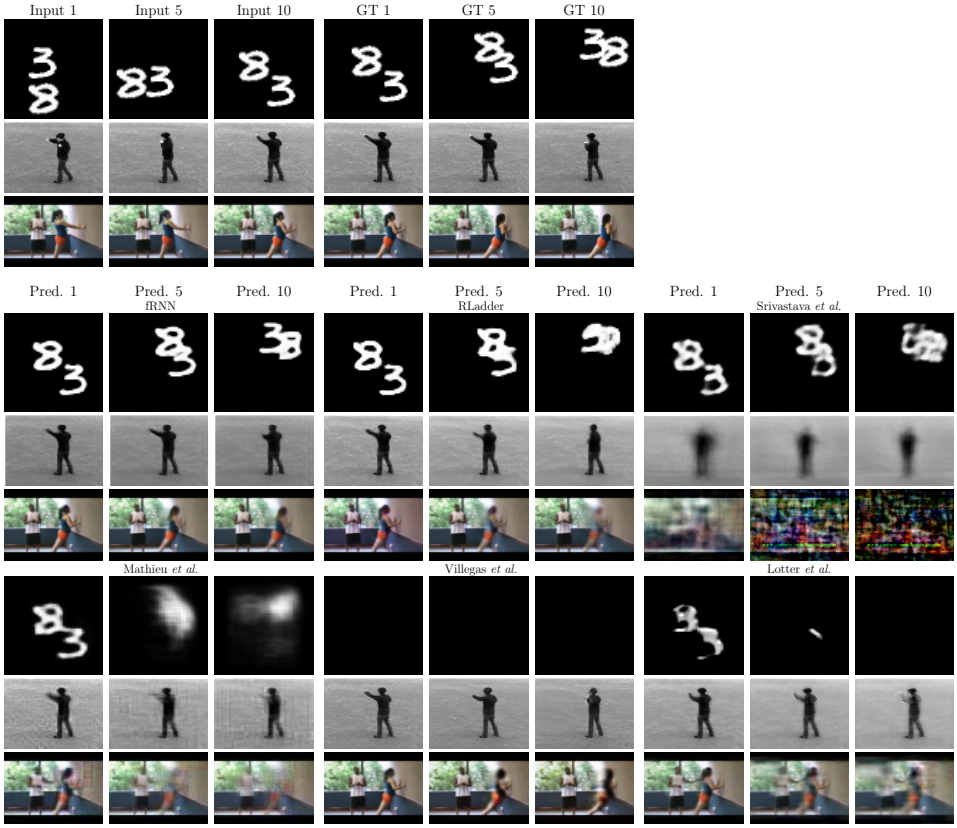



Figure 4.6: Predictions at 1, 5, and 10 time steps from the last ground truth frame. RLadder predictions on MMNIST are from the model pre-trained on KTH.

both accumulate blur over time, but while the former does so to a smaller degree, the latter makes the digits unrecognizable after five frames.

For KTH, Villegas *et al.* obtains outstanding qualitative results. It predicts plausible dynamics and maintains the sharpness of both the individual and background. Both fRNN and RLadder follow closely, predicting plausible dynamics, but not being as good at maintaining the sharpness of the individual. On UCF101,

our model obtains the best predictions, with little blur or distortion compared to the other methods. The second best is Villegas *et al.*, successfully capturing the movement patterns but introducing more blur and important distortions on the last frame. When looking at the background, fRNN proposes a plausible initial estimate and progressively completes it as the woman moves. On the other hand, Villegas *et al.* modifies already generated regions as more background is uncovered, producing an unrealistic sequence. Srivastava *et al.* and Lotter *et al.* fail on both KTH and UCF101. Srivastava *et al.* heavily distort the frames. As discussed in Section 4.4.3, this is due to the use of fully connected recurrent layers, which constrains the state size and prevents the model from encoding relevant information on complex scenarios. In the case of Lotter *et al.*, it makes good predictions for the first frame, but rapidly accumulates artifacts.

Additional qualitative results are found in Figures 4.8, 4.9, 4.10 for MMNIST, KTH and UCF-101 respectively. These are compared against all considered models, except for Villegas *et al.* in the case of MMNIST, where the method does not converge.

### 4.4.5 Representation stratification analysis

Here we analyse the stratification of the sequence representation among dGRU layers. Because dGRU units allow for a bidirectional mapping between states, it is possible to remove the deepest layers of a trained model in order to check how the predictions are affected, providing an insight on the dynamics captured by each layer. To our knowledge, this is the first topology allowing for a direct observation of the behaviour encoded on each layer.

In Figure 4.7, the same MMNIST sequences are predicted multiple times, removing a layer each time. The analyzed model consists of 2 convolutional layers and 8 dGRU layers. Firstly, removing the last 2 dGRU layers has no significant impact on prediction. This shows that, for this dataset, the network has a higher capacity than required. Further removing layers results in a progressive loss of behaviors, from more complex to simpler ones. This means information at a given level of abstraction is not encoded into higher level layers. When removing the third deepest dGRU layer, the digits stop bouncing at the edges, exiting the image. This indicates this layer encodes information on bouncing dynamics. When removing the next one, digits stop behaving consistently at the edges: parts of the digit bounce while others keep the previous trajectory. While this also has to do with bouncing dynamics, the layer seems to be in charge of recognizing digits as single units following the same movement pattern. When removed, different segments of the digit are allowed to move as separate elements. Finally, with only 3-2 dGRU layers the digits are distorted in various ways. With only two layers left, the general linear dynamics are still captured by the model. By leaving a single dGRU layer, the linear dynamics are lost.

According to these results, the first two dGRU layers capture pixel-level movement dynamics. The next two aggregate the dynamics into single-trajectory components, preventing their distortion, and detect the collision of these components with image bounds. The fifth layer aggregates single-motion components into digits, forcing them to behave equally. This has the effect of preventing bounces, likely due to only one of the components reaching the edge of the image. The sixth dGRU layer provides coherent bouncing patterns for the digits.

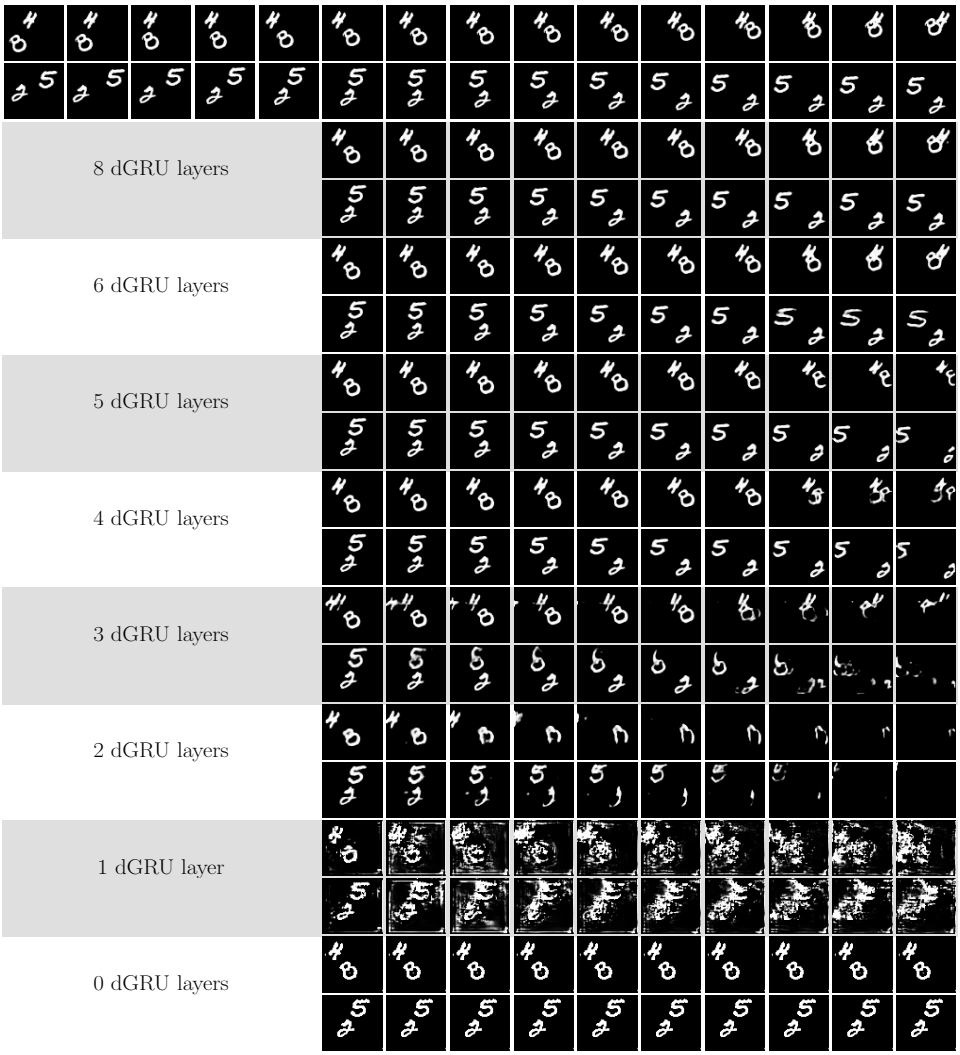



Figure 4.7: Moving MNIST predictions with fRNN layer removal. Removing all dGRU layers (last row) leaves two convolutional layers and their transposed convolutions, providing an identity mapping.

## 4.5 Conclusions

We have presented Folded Recurrent Neural Networks, a new recurrent architecture for video prediction with lower computational and memory costs compared to equivalent recurrent auto-encoder models. This is achieved by using the proposed double-mapping GRUs, which horizontally pass information between the encoder and decoder. This eliminates the need for using the entire auto-encoder at any given step: only the encoder or decoder is executed for both input encoding and prediction, respectively. It also facilitates the convergence by naturally providing a noisy identity function during training. We evaluated our approach on three video datasets, outperforming state-of-the-art techniques on MMNIST and UCF101, and

obtaining competitive results on KTH with 2 and 3 times less memory usage and computational cost than the best scored approach. Qualitatively, the model can limit and recover from blur by preventing its propagation from low to high level dynamics. We also demonstrated stratification of the representation, topology optimization, and model explainability through layer removal. Layers have been shown to successively introduce more complex behaviors: removing a layer eliminates its behaviors but leaves lower-level ones untouched.

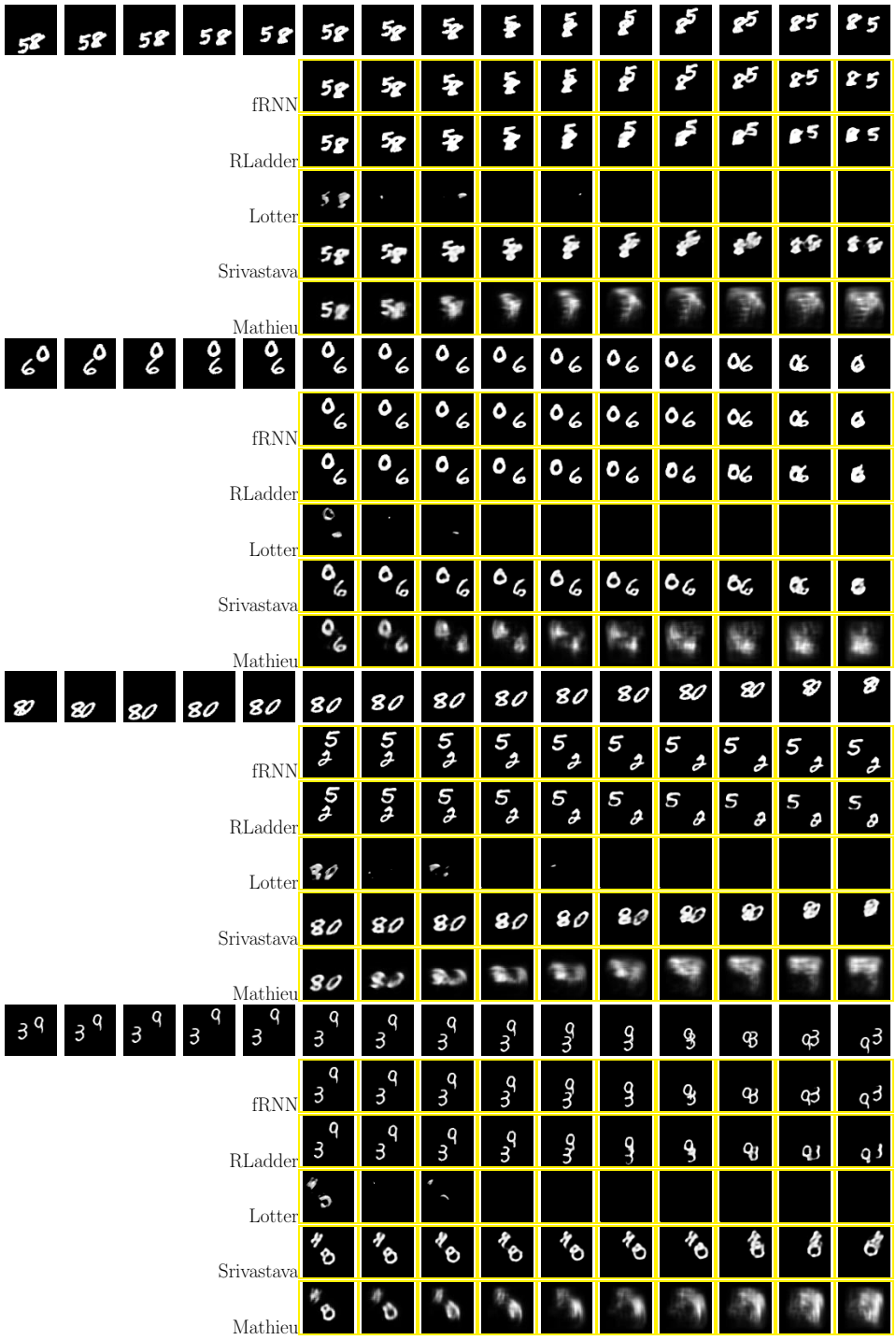



Figure 4.8: fRNN predictions on MMNIST. First row per sequence shows last 5 inputs and target frames. Yellow means prediction.

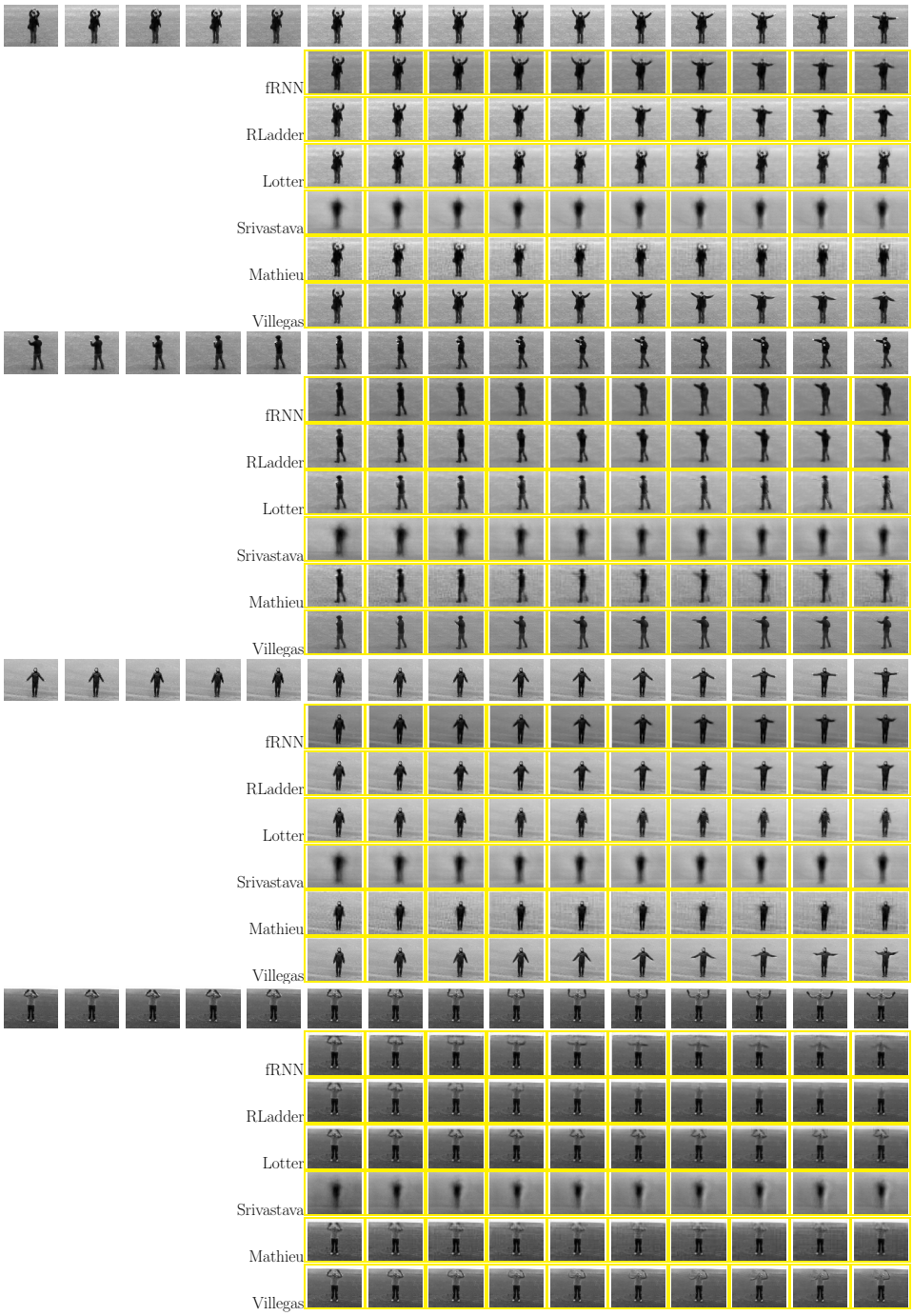



Figure 4.9: fRNN predictions on KTH. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

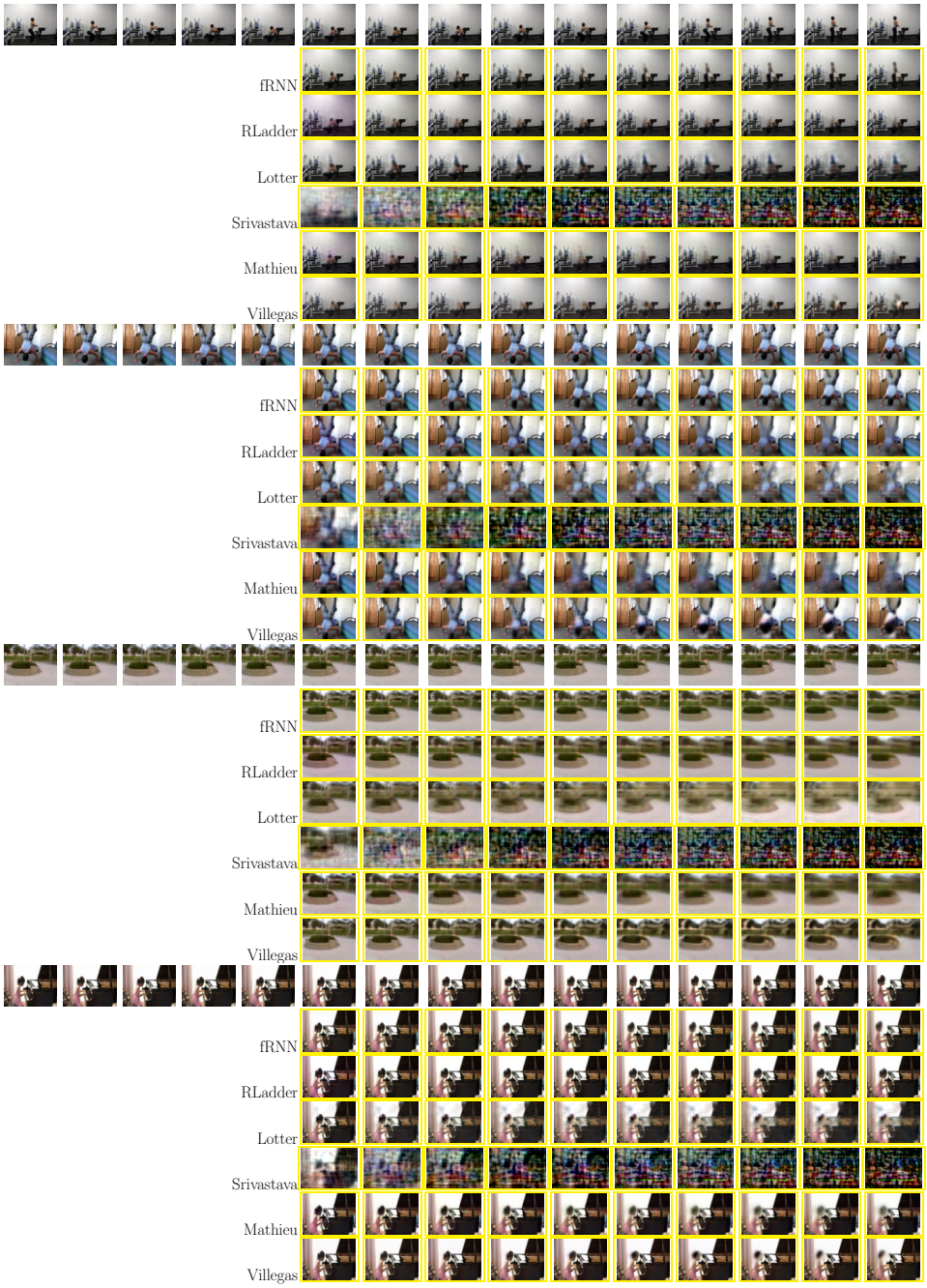



Figure 4.10: fRNN predictions on UCF101. First row for each sequence shows last 5 inputs and target frames. Yellow frames are model predictions.

# Part III

# Cumulative Distribution Estimation

# Chapter 5

# Multi-varied Cumulative Alignment

## 5.1 Introduction

Domain Adaptation (DA) is a sub-domain of transfer learning focused on applying an algorithm trained in one or more source domains to a related target domain. Both domains share the same feature space, but the sample distribution in that space is different. The distribution discrepancy, which commonly entails some kind of affine transform of the feature space, is known as the domain shift. The goal of DA is to find and eliminate the domain shift, such that, once the algorithm is applied, the method originally trained on the labeled source domain increases its accuracy on the target domain. One example may be the difference between hand-written and computer-generated characters. While both of them share the same basic feature space, it will inevitably shift due to the slightly different appearance between images in both domains.

DA approaches can be classified into two main families: non-parametric and parametric. While both model the probability distribution of the samples, they do so in two different ways. Non-parametric methods compute statistical indicators such as feature-space covariances in order to approximate the PDF of each domain, while parametric methods use learnable parameters, usually in the form of neural networks, to create a more complex model of the distribution. These two families and their methods are explained in Section 5.2.

In this work, we propose a new non-parametric method which is capable of reducing the domain shift between complex probability distributions. Differently from all other non-parametric methods, it takes into account the complex internal structure of the distribution, while still retaining the main advantages of such approaches: no learnable parameters and a minimal memory footprint. This is done by minimizing, through an auxiliary loss, the discrepancy between random Cumulative Distribution Function (CDF) estimations as measured in both the source and target domains feature spaces. In order to do away with problems caused by the high dimensionality of the space (curse of dimensionality), a series of 1-dimensional random projections of the probability space are used instead. An overall alignment of the non-collapsed PDFs is obtained by using this metric on a high number of both random points of the space and directions of projection.

## 5.2 Related work

There are two main ways of tackling a Domain Adaptation model. The first, is to use statistical indicators and techniques to model the either probability distributions of the source and target domains, or the statistical discrepancy between both. This then allows us to either transform the target domain sample features so that they follow the source domain distribution, or to define an auxiliary loss that minimizes the domain shift during training. The family of methods following this approach could be considered the more classical ones, many of them predating the popularization of deep learning. The following are some of the main approaches belonging to this family.

Sun *et al.* propose CORrelation ALignment (CORAL) [74], a simple technique whose purpose is to bring the target domain feature representations of an already trained model to those of the source. This is done by using the covariance matrices of both domains as a statistical measure of their discrepancy, whitening the target features and then re-coloring them by using the source covariance. This is easily achieved by using the formula $\tilde{z}_S = z_S C_S^{-\frac{1}{2}} C_T^{\frac{1}{2}}$, where $z_s$ is the feature representation of a source sample, and $C_{S/T}$ are the source and target covariances. Using such an approach means that a transform $S = C_S^{-\frac{1}{2}} C_T^{\frac{1}{2}}$ must be learned and applied at test time for each pair of source and target domains.

An extension of this method called Deep CORAL [75], also proposed by Sun *et al.*, bypasses this problem by jointly minimizing the target loss alongside the hidden representation covariances by minimizing the squared Frobenius norm of the difference in covariance matrices. This results in a single-step training process that does not need to consider the domain shifts at test time.

MK-MMD minimization [76] introduced by Long *et al.*, similarly to Deep CORAL, minimizes the discrepancy between target hidden representations through the use of an auxiliary loss. In this case, a multiple kernel maximum mean discrepancy (MK-MMD) [77] is used instead of the domain covariances.

The above methods suffer from precision problems during modeling. For instance, it is easy to see that a method such as CORAL will approximate the source and target distributions to multi-variate Gaussians. This disregards any kind of internal structure the distributions may display, effectively aligning only the overall outline instead. In order to accurately represent the domain distributions, it becomes necessary to use a higher order model. More modern approaches solve this by using auxiliary deep neural networks which, either directly or indirectly, model the sample distributions of both domains.

Deep Adaptation Networks (DAN) [25], put forward by Ganin *et al.* in 2014, was the first to propose using adversarial learning in order to minimize the domain discrepancy. The method consists of a neural feature extractor, whose features are then fed into two neural classifiers; a class classifier that predicts the sample class, and a domain classifier which predicts whether the sample comes from the source or target domain. Instead of a two-step adversarial training, the authors introduce the Gradient Reversal Layer (GRL). This layer acts as an identity operator during the forward pass and inverts the gradients during back-propagation. Placing it in-between the feature extractor and domain classifier results on an overall model where domain discrepancies are adversarially minimized alongside the class classification accuracies. Essentially, the domain classifier is trained to distinguish between both domains, while the feature extractor reduces the domain shift in order

to fool the classifier.

Bousmalis *et al.* proposed Deep Separation Networks (DSN) [26] in 2016. This is an extension to DANs where, in addition to a class classifier and an adversarial domain classifier, a stratification of the feature representation is proposed. More specifically, the model uses three feature extractors: one extracts features common to both domains, while the other two are domain-specific. The class and domain classifiers work solely on the common features, while an auxiliary convolutional network reconstructs the input images based on the concatenation of the common and domain-specific features, ensuring that the full representation contains all of the information in the image.

Saito *et al.* [27] were the first to propose an adversarial approach that explicitly takes into account, to dome degree, the internal structure of the probability distributions for domain adaptation. To do so, they align not only the PDFs of both domains, but also the classification decision boundaries. Instead of the typical architecture consisting of a feature extractor and domain and class classifiers, this approach replaces the domain classifier by a second class classifier. The training process is then performed in two steps. On the first, the feature extractor and both class classifiers are minimized to fit the source domain samples. On the second, the two class classifiers are trained to maximize their discrepancy on the target domain class predictions while the feature extractor is trained to minimize that same discrepancy. This training procedure brings the target domain feature space closer to to that of the source domain, the samples of the latter serving as supports defining the valid regions of the feature space.

In a similar fashion, Lee *et al.* [28] propose using the same type of architecture and multi-step training approach as in [27]. Differently from them, they use the Sliced Wasserstein Distance (SWD) as the distance metric to determine the similarity between the target domain samples in both classifiers. This more statistically sound metric to measure the similarity between class probability distributions results in a higher accuracy on the target domain, boosting the final target classification accuracy by about 3% in most datasets.

Other works, such as ADA-DM by Xu *et al.* [78], propose alternative variants of adversarial training. Here, a neural network generates a feature distribution for each sample, consisting of a mean value and standard deviation for each feature. This allows the authors to both directly evaluate the source samples with a classification network, as well as combining source and target distributions to create a new mixed domain generator. An alternative generator-discriminator pair of modules is then used to generate and evaluate samples of both the source, target, and mixed domains, providing an additional classification loss, as well as both adversarial domain classification and custom triplet learning losses.

More recent approaches propose combining both metric learning with adversarial modeling and minimization of the probability distributions. This is the case of Maximum Density Divergence (MDD) by *et al.* [79]. In said work, a new statistical metric is used to minimize both the inter-domain divergence and maximize the intra-class density of both domains feature spaces based on predicted target domain labels. Additionally, a domain classifier over the label probability distributions is used as an adversarial loss to minimize domain discrepancies.

We have seen that non-parametric methods such as CORAL and MK-MMD more directly align the source and target PDFs through a series of statistical measurements on the batch samples. While doing so does not require additional train-

ing parameters, it only allows for simple (quadratic) models of the PDFs. On the other hand, parametric approaches use additional neural modules to produce deep models of the PDFs either directly, like DAN and DSN, or indirectly, like MCD and SWD. In contrast, we propose a non-parametric approach with a new type of statistical measurement that does take into account the internal complexity of the PDFs, bringing together the best of both worlds.

## 5.3 Proposed method

Classic DA approaches based on distribution alignment treat the PDFs of the source and target domains as multivariate Gaussians that are aligned by matching statistical measures, e.g. mean and covariance. While this works to some degree, any information regarding the internal structure of the distributions is lost. This may cause a few problems when trying to reduce the domain shift:

1. Due to domain shift, the main covariance directions and order may be different between domains, resulting in a misalignment. This is especially true for lower covariance components, which generally encode less relevant information but can still have an impact on the end result of a classifier.
2. Not taking into account the internal structure may result in a generally correct alignment of the domains in terms of their major variance directions, but with the internal structure being distorted either through stretching, compression or twisting of certain regions.

These problems are shown in Figure 5.1, which showcases a simple moons dataset two-class classification problem. The domain shift between both domains (left), causes the Probability Distribution Functions (PDFs) to not match each other (right). If we were to compute the maximum variance axes on each domain and use that to remove the covariance shift, we would end up with matching Gaussian distributions, but the internal structure of both domains would be misaligned, resulting in a high classification error.

In order to solve this, we indirectly align the PDF through its Cumulative Distribution Function (CDF). A straightforward approach to estimating the CDF evaluated at a given location is measuring the number of points in the batch that fall below the coordinates of that point for all dimensions of the representation or, what is the same, fall on the negative octant relative to the evaluated location. This, though, suffers from a big problem: As the dimensionality of the representation increases, the fraction of the latent space falling on the negative octant also quickly decreases, leaving us with few or no samples to get a precise estimate of the CDF. We propose to circumvent this problem by working on random 1-d projections of the feature space, as explained in Section 5.3.2.

The proposed DA approach consists of two steps, as seen in Figure 5.2, where the full training pipeline is shown. Firstly, we propose a new algorithm that finds a re-weighting of the source samples such that the source covariance matches that of the target. This allows us to re-balance both domains, mitigating problems related to relative imbalances of the data between domains, such as class imbalance. Secondly, we propose a new approach for CDF alignment that avoids the problem of high dimensionality feature spaces.

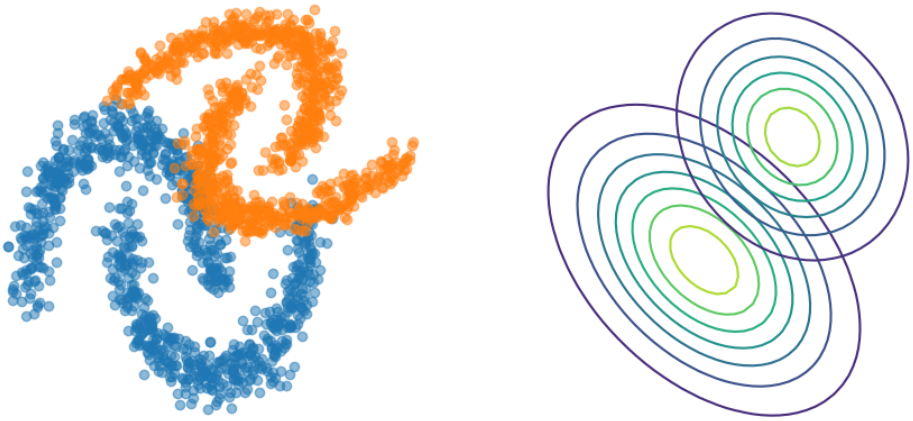


Figure 5.1: Toy example of the source and target domain distributions of a two-class classification toy problem (moons dataset). Left: samples in the source (blue) and target (orange) domains, with a domain shift that includes rotation, translation and skewing. Right: Probability Distribution Functions of both domains.

### 5.3.1 Balancing sample distributions

When aligning the Probability Distribution Functions (PDFs) $P_s(x_i)$ and $P_t(x_i)$, it is important to have similar sample representations of the functions. In the case of a classification task, for instance, class imbalances between source and target domains would lead to different PDFs, hindering the alignment process. Yet, not knowing the target labels during training prevents us from balancing the datasets. To address this, we propose estimating a weighting of the source samples that increases the contribution of samples in under-represented areas of the PDF, re-balancing the source domain such that it more closely matches the target PDF. We use the source and target covariances as a proxy to obtain these weights, with the goal of matching the target covariance by weighting the source samples. Given $\widetilde{X}_S = \{x_i^{(s)} - \overline{x}^{(s)}\}$ and $\widetilde{X}_T = \{x_i^{(t)} - \overline{x}^{(t)}\}$, our goal is:

$$\begin{aligned} &\arg\min_w ||\frac{1}{N_S}\widetilde{X}_S^T D_w \widetilde{X}_S - \frac{1}{N_T}\widetilde{X}_T^T \widetilde{X}_T||_F^2 \\ &= \frac{1}{N_S^2} w^T (\widetilde{X}_S \widetilde{X}_S^T)^{\circ 2} w - \frac{2}{N_S N_T} \mathbb{1}^T (\widetilde{X}_T \widetilde{X}_S^T)^{\circ 2} w + \frac{1}{N_T^2} \mathbb{1}^T (\widetilde{X}_T \widetilde{X}_T^T)^{\circ 2} \mathbb{1} \end{aligned} \tag{5.1}$$

Here, $D_w$ is the diagonal matrix whose entries are the source sample weights, while $N_S$ and $N_T$ correspond to the number of source and target samples. Thus, this equation aims to find the weights $w$ minimizing the squared Frobenius norm between the source and target covariances. This problem can be solved with the next closed form solution, where $\circ$ denotes the Hadamard power:

$$w = \left(\frac{1}{N_S}\left(\widetilde{X}_S \widetilde{X}_S^T\right)^{\circ 2}\right)^{-1} \frac{1}{N_T}\left(\widetilde{X}_S \widetilde{X}_T^T\right)^{\circ 2} \mathbb{1}^{<N_T \times 1>} \tag{5.2}$$

Note that we defined the centered source samples as $\widetilde{x}_i^{(s)} = x_i^{(s)} - \overline{x}^{(s)}$. This is not accurate, since the mean changes with the weighting $w$ such that $\widetilde{x}_i^{(s)} =$

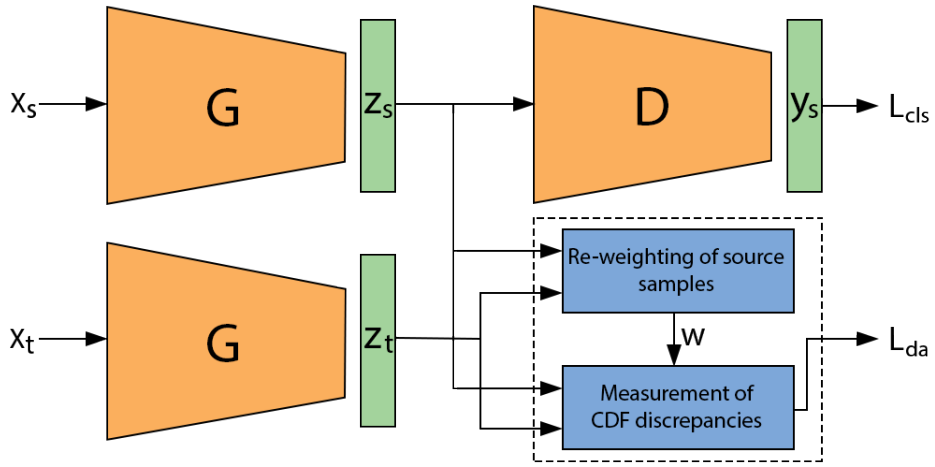



Figure 5.2: General training pipeline. A feature extractor $G$ generates features for both the source and target domain samples in the training batch. Source samples are passed through the discriminator $D$ and evaluated against the source targets using the classification loss $L_{cls}$. Domain Adaptation is done through a trwo-step process, shown inside the dashed box. First a re-weighting for the source samples is found to correct for potential class imbalances between both domains. The CDF discrepancies between both domains are then estimated at various points and minimized through the Domain Adaptation loss $L_{da}$.

$x_i^s - \frac{1}{N_S} w^T X_S$. Plugging this into Equation 5.1 would lead to a quadratic form for $w$, forcing us to resort to quadratic optimization techniques. Instead of that, we initially assign an equal weight $w_i = \frac{1}{N_S}$ to each sample, then iteratively compute the centered sample features and re-estimate the weights using Equation 5.2, in a fashion similar to Procrustes analysis. This is shown in Alg. 1.

Please note that we perform an L1 normalization of the sample weights at the end of Alg. 1. This is in order to have an equivalent number of samples in both domains after the weighting of source samples.

### 5.3.2 Aligning the Cumulative Distribution Functions

In order to align the source and target CDFs $C_s(x)$ and $C_t(x)$, we will want to, given a batch of samples, estimate the value of these functions at random points of the distribution and minimize the difference across domains. Unfortunately, this cannot be directly done for high-dimensional feature spaces such as those obtained as the output of a hidden layer. This is due to the number of samples falling into the negative octant of that space being statistically insignificant, which is commonly known as the curse of dimensionality. The number of samples needed to obtain a statistically significant estimation of the CDF is in the order of $\mathcal{O}(2^d)$, where $d$ is the space dimensionality.

In order to side-step this issue, we propose evaluating the CDF at random cut-off points within random 1-d projections of both functions. Each such random projection now has a statistically significant number of samples to each side of any chosen cut-off point on the line and, through many such random projections, we

**Algorithm 1** Weighting of source samples

$\widetilde{X}_T \leftarrow X_T - \frac{1}{N_T}\mathbb{1}^{<1\times N_T>}X_T$
$w \leftarrow \mathbb{1}^{<N_S\times 1>}$
**for** *iter* **in** $[0 \ldots k]$ **do**
    $\widetilde{X}_S \leftarrow X_S - \frac{1}{N_T}w^T X_S$
    $A \leftarrow \frac{1}{N_S}(\widetilde{X}_S\widetilde{X}_S^T)^{\circ 2}$
    $B \leftarrow \frac{1}{N_T}(\widetilde{X}_S\widetilde{X}_T^T)^{\circ 2}$
    $w \leftarrow A^{-1}B\mathbb{1}^{N_T\times 1}$
**end for**
$w \leftarrow \frac{N_S}{|w|_1}w$

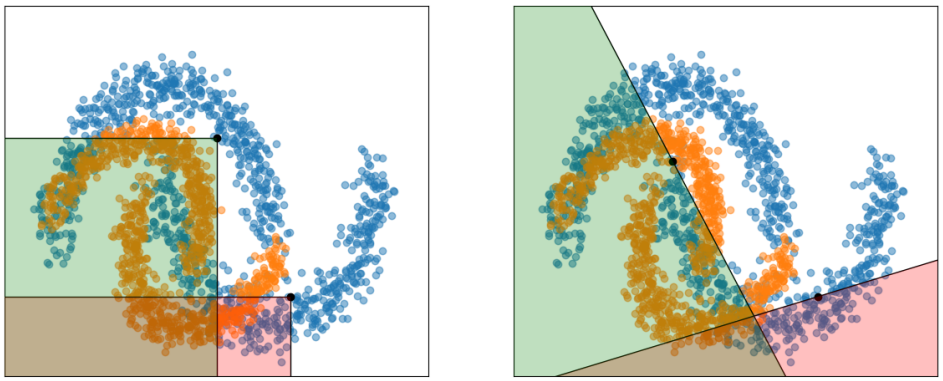


Figure 5.3: Left: The CDF estimate at a given point corresponds to the fraction of samples that fall on the negative octant relative to the point. Right: Proposed projection method, where a random hyper-plane containing the point cuts the space in two, effectively estimating the CDF on a random 1-dimensional projection of the space.

are able to fully map all orientations in the feature space. A simple comparison between regular CDF estimation and the proposed approach is shown in Figure 5.3.

Lets define two functions $C_s(x, v)$ and $C_t(x, v)$ which take as input a given cut-off point $x \in \mathbf{R^d}$ for which the CDF is evaluated, and a vector $v \in \mathbf{R^d}$ over which function $C_{\{s,t\}}(x)$ is projected. Then $C_{\{s,t\}}(x, v)$ corresponds to the probability of a sample falling to the negative side of the hyper-plane with normal $v$ and containing the point $x$. The estimate $\widetilde{C}_s(x, v)$ of such a function corresponds to the following equation:

$$\widetilde{C}_s(x, v) = \frac{1}{N_s}\delta\left(X_s v - x^T v\right)\mathbb{1}^{<N_s\times 1>} \tag{5.3}$$

Here, $\delta(x)$ is the step function taking the value 1 for negative values of $x$ and 0 otherwise. Optimizing this function through back-propagation is not possible due to $\delta(x)$ not being differentiable. Instead, we propose using a sigmoid function to approximate it. We will also want to use the sample weighting defined in Section 5.3.1 when computing $\hat{C}_s(x, v)$. This is done to find a CDF estimate over the re-balanced source dataset, which more closely matches the CDF of the target. This

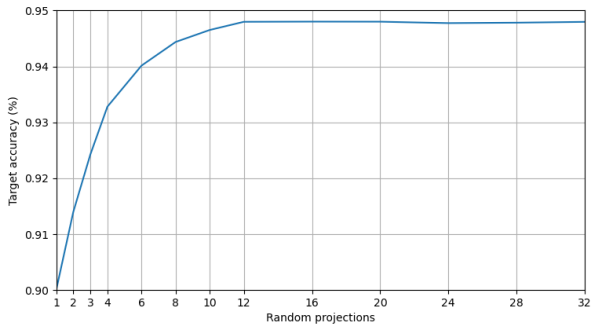



Figure 5.4: Calibration of parameter $n_p$ on the SVHN to MNIST domain adaptation problem. Each value corresponds to the average accuracy over 32 runs, relative to the number of random 1-dimensional projections of the feature space per sample.

gives the following pair of equations:

$$\begin{aligned} \hat{C}_s(x,v) &= \frac{1}{N_S}\sigma\left(X_S v - x^T v\right) w \\ \hat{C}_t(x,v) &= \frac{1}{N_T}\sigma\left(X_T v - x^T v\right) \mathbb{1}^{<N_T \times 1>} \end{aligned} \tag{5.4}$$

Note that these functions approximate the probability of a sample falling on the *positive* side of the hyper-plane instead of the negative one. Since the projection vector $v$ is picked at random and is as likely to point in one direction than the other, this does not affect the method. In order to match the projected CDF estimates of both domains, we use entropy maximization over the normalized estimates. Given a set of random hyper-planes $P = \{(x_1, v_1),\ ..,\ (x_k, v_k)\}$, the domain adaptation loss $L_{da}$ is:

$$\begin{aligned} L_{da} &= \frac{1}{k}\sum_{(x,v)\in P} \overline{C}_s(x,v) log(\overline{C}_s(x,v)) + \overline{C}_t(x,v) log(\overline{C}_t(x,v)) \\ \overline{C}_s(x,v) &= \frac{\hat{C}_s(x,v)}{\hat{C}_s(x,v) + \hat{C}_t(x,v)}, \quad \overline{C}_t(x,v) = \frac{\hat{C}_t(x,v)}{\hat{C}_s(x,v) + \hat{C}_t(x,v)} \end{aligned} \tag{5.5}$$

Note that multiple random projection vectors $v$ can be used for each sample when evaluating $\overline{C}_s(x,v)$, obtaining multiple evaluations of the DA loss functions per sample. This allows us to map the probability space without the need to increase the batch size. We use this during training, with hyper-parameter $n_p$ defining the number of random 1-dimensional space projections per sample.

Calibration is conducted for $n_p$ in Figure 5.4 using the SVHN to MNIST task (see Section 5.4 for more information). As shown, the accuracy gradually increases with the number of random feature space projections per sample, until it plateaus at around $n_p = 12$. We obtain similar results for the other datasets. As such the parameter is set to 12 for all experiments.


Figure 5.5: Samples from the three digits datasets. They consist of ten classes for the digits from 0 to 9. MNIST is the simplest of them, the images having been processed to display full contrast (each pixel is either completely black or white). USPS is similar to MNIST, but with images not having been cleaned up. This results in heavy variations in contrast, as well as some level of blurriness and changes in intensity across the digit's stroke. SVHN consists of home numbers captured from the street, with varying typographic styles and containing distractors in the form of neighboring numbers.

### 5.3.3 Training loss

The training loss is obtained by linearly combining the classification loss $L_{cls}$ and Domain Adaptation loss $L_{da}$. The first is a regular cross-entropy loss, commonly used in classification problems, while the latter is obtained as shown in Equation 5.5. The second loss is weighted by a tunable parameter $\lambda$, as shown below.

$$L = L_{cls} + \lambda L_{da} \tag{5.6}$$

The weighting parameter $\lambda$ is obtained through annealing, with a value from 0 growing through the training process to asymptotically converge to $\Lambda$. This is done to give time to converge to a useful feature representation using the source domain samples, before gradually enforcing that representation to match that of the target domain. The value of $\lambda$ is obtained as below, where $t$ is the current mini-batch training iteration.

$$\lambda = \left( \frac{2}{1 + e^{-\frac{t}{1000}}} - 1 \right) \Lambda \tag{5.7}$$

Please note that this function follows a sigmoid curve centered at zero. Its value steadily increases, eventually converging at 1, at which point $\lambda = \Lambda$. In all of our experiments, we used $\Lambda = 100$.

## 5.4 Experiments

We focus on four common Domain Adaptation classification tasks. We used the same proposed topologies for the classification models, changing only the DA loss, and the same training/testing data partitions. We note that our method does not work properly when the DA loss is applied to the output of a convolutional layer. As such, while the network topology has remained intact in every case, the DA loss has been moved to an appropriate layer when necessary, resulting in slightly different feature extraction/classification partitions. The datasets, data partitions, as well as the topology used for each task, are explained below.


Figure 5.6: Samples from both traffic sign classification datasets. GTSRB (top): it consists of real world images under different illumination conditions and minor obstructions in some cases. SYNSIG (bottom): Synthetic dataset, with the signs drawn on top of random backgrounds. Blur, lightning, color saturation and geometric distortions are randomly introduced to the samples.

**SVHN → MNIST.** Here the classification model is trained on using SVHN as a source dataset, which consists of images taken from house numbers. We use 73257 images for training and 26032 for testing. The target dataset is MNIST, which consists of hand-written numbers between 0 and 9. We use 55000 images for training and 10000 for testing. The images are resized to $32 \times 32$ pixels for both datasets. The feature extractor consists of three convolutional layers with convolutions of size $5 \times 5$, respectively with 64, 64 and 128 convolutional kernels, and a fully-connected layer with 3072 hidden units. All layers are followed by batch normalization, and in the case of the convolutional layers, a $2 \times 2$ max. pooling. The feature extractor consists of a dropout layer with $p = 0.5$ followed by two fully-connected layers with 2048 and 10 neurons respectively.

**MNIST ↔ USPS.** These are two similar datasets displaying hand-written digits from 0 to 9, for a total of ten classes. In the case of MNIST, we use 55000 images for training and 10000 for testing. For USPS, we have 7438 and 1860 train and test samples, respectively. In both cases the images are resized to $28 \times 28$ pixels. The feature extractor consists of two $5 \times 5$ conv. layers with 32 and 48 kernels respectively, followed by a fully-connected layer with 100 hidden units. The convolutional layers are followed by $2 \times 2$ max. pooling layers, and all layers implement batch normalization. The classifier consists of two fully-connected layers with 100 and 10 hidden units respectively, each trailed by a drop-out layer with $p = 0.5$.

**SYNSIG → GTSRB.** These two datasets display 43 common traffic signs.

| | SVHN to MNIST | SYNSIG to GTSRB | MNIST to USPS | USPS to MNIST |
|---|---|---|---|---|
| Source only | 67.1% | 85.1% | 79.4% | 63.4% |
| Non-parametric Distribution Matching methods | | | | |
| MMD [76] | 71.1% | 91.1% | 81.1% | - |
| MCA (Ours) | **94.8**% | **95.1**% | **94.6**% | **94.7**% |
| Parametric Distribution Matching methods | | | | |
| DAN [25] | 71.1% | 88.7% | 85.1% | 73.0% |
| DSN [26] | 82.7% | 93.1% | - | - |
| MCD [27] | 96.2% | 94.4% | 96.5% | 94.1% |
| SWD [28] | **98.9**% | **98.6**% | **98.1**% | **97.1**% |
| DM-ADA [78] | 95.5% | - | 96.7% | 94.2% |

Table 5.1: Classification accuracy of various DA models on the four most common Domain Adaptation tasks. Results on the other methods are obtained from their respective papers. While the proposed approach is surpassed by SWD, it by far outperforms any non-parametric approaches, obtaining results comparable to or even better than most parametric methods.

SYNSIG, the source dataset, consists of synthetic images. We use 100000 images for training and 2000 for testing. GTSRB, the target dataset, consists of real-world images taken in Germany. We use 31367 images for training and 32171 for testing. In both cases the images are resized to $40 \times 40$ pixels. The feature extractor consists of three convolutional layers with kernels of size $5 \times 5 \times 96$, $3 \times 3 \times 144$ and $5 \times 5 \times 256$, and a fully-connected layer with $512 hiddenunits$. All layers are followed by a batch normalization and, in the case of the convolutional layers, a $2 \times 2$ max. pooling. The classifier consists of a single fully-connected layer with 43 neurons, trailed by a $p = 0.5$ dropout layer.

Figure 5.5 displays visual samples of the three digit classification datasets, namely MNIST, USPS and SVHN, displaying the domain differences between the three datasets. For the traffic sign datasets, samples are shown in Figure 5.6.

Table 5.1 shows a comparison of the state-of-the-art methods against our proposed approach. Methods are divided into classic non-parmetric distribution matching, and parametric approaches based on the usage of additional neural modules plus adversarial training. As it can be seen, our approach far surpasses the accuracy of non-parametric methods on all datasets, despite belonging to that same category. It also outperforms many of the parametric methods, such as DAN [25] and DSN [26], and surpasses MCD [27] for both SYNSIG to GTSRB and USPS to MNIST. It also surpasses DM-ADA for the latter. The only approach that consistently surpasses our method is SWD [28], which has a classification accuracy 3% higher in the average case.

This is despite our approach being non-parametric, meaning that it does not require additional neural modules nor a multi-stage training approach. Overall, our method achieves accuracies comparable to the state-of-the-art without additional training parameters and with a smaller computational footprint.

## 5.5 Conclusions

We proposes a new DA approach that does not require additional training parameters. It bypasses the need for both additional neural modules and multi-stage training by providing an efficient metric for the CDF alignment of both the source and target feature spaces. This results in lower memory requirements, a simpler training pipeline and the ability to consider the internal structure of the probability distributions, all without needing to model the distributions themselves. While our approach does not achieve state-of-the-art results, it does provide accuracies that are very close to them, just 3% lower than the best approach. At the same time, it does so without requiring additional training steps or trainable parameters, making use of a new loss that directly minimizes the domain shift. This is in line with the workings of classical non-parametric distribution matching methods, compared to which our approach greatly surpasses their accuracy. Furthermore, our model could be directly applied to any Domain Adaptation problem, regardless of it being a regression or classification task. This is demonstrated by the fact that Domain Adaptation is performed on the latent feature space instead of the label space, and it does not depend on the classification labels whatsoever, contrary to the all state-of-the-art approaches such as [27, 28]. We consider testing such methods on other regression-based tasks as interesting future work.

# Chapter 6

# Final discussion and conclusion

Since the popularization of GPUs in machine learning, and with it neural networks, many Computer Vision tasks has benefited from the introduction of ever more complex models, requiring both more memory and computational power. In this work, we have seen that this is not always necessarily the best solution, and that by taking into account the nature of both the task and the algorithm, other more efficient solutions can be found. We have seen three different task-specific approaches to modeling high order behavior. In every instance, we have taken a step back and looked at the problem from a different perspective, proposing highly efficient solutions that either surpass the state-of-the-art or are comparable to it, while requiring less memory and computational resources.

In the case of face alignment, where the literature focused on cascaded linear regressors to iteratively approximate an initial facial geometry to the real location of the landmarks in the image, the main obstacle was the treatment of images with heavy rotations and illumination changes. The state-of-the-art proposed two solutions to such a problem. Either use multiple random initializations at test time and then look for a converging geometries [37, 33, 2, 3], or partition the feature space along the directions of maximum variance and define an independent regressor for each sub-region [24]. We instead focused on increasing the complexity of the regressor itself, defining a second order regressor where second order relationships are taken into account only relative to the maximum variance variables. This makes a much more efficient use of the subspace compared to GSDM [24]. By not partitioning the space, we have a single regressor that can be trained on the full dataset, while the total number of training variables is greatly reduced and the model's flexibility increases. This additionally eliminates the need for multiple test-time executions with different initializations.

We then moved out of more classical machine learning approaches and into deep learning, first analyzing the ability of a neural network to capture high order relationships in the input data relative to the depth of the network. The first deep learning task we worked on was future video prediction, commonly achieved through the use of some flavor of recurrent neural networks. Making use of the insights on the order of complexity of the encoded information relative to the network's depth, we put forward a new auto-encoder topology where the states are fully shared

between encoder and decoder, with the corresponding modifications to the training pipeline. This has several advantages over the state of the art. First, lower order information does not need to go through higher order layers, reducing the overall number of parameters in the neural network by eliminating unnecessary encoding of information. Second, because of the fully shared states, these can be updated either through the encoder or decoder, without needing to use both consecutively. This means that, both during training and execution, only half of the topology needs to be executed, further halving both the memory and computational requirements. Another advantage of this is the elimination, to a high degree, of image degradation: recurrent approaches typically need to feed the decoder's prediction back as input in order to update the recurrent states before making the following prediction. This re-introduces any prediction mistakes and magnifies them in subsequent predictions. By eliminating the need to re-encode the inputs, this problem disappears.

The third task we have seen is that of Domain Adaptation (DA), where the shift in the probability distribution of the between two domains' feature spaces is minimized in order to allow a model trained in one of the domains to work on the other. There, we have seen how in recent years DA approaches have shifted to adversarial-based models, where additional neural modules capture the complexity of the probability distributions for later minimization. Contrary to the state of the art, he have taken a step back and found a different approach, more aligned with classical domain adaptation in that it works based off metric alignment. To do so, we have defined a new metric based on estimations of the Cumulative Density Function (CDF) of both distributions, allowing us to capture the internal structure of high order probability distributions without having to resort to modeling the distributions themselves. This brings together the best of both worlds. As with classical metric based approaches, we do not need additional trainable parameters nor depend on multi-stage training approaches. As with more modern adversarial approaches, we take into account the internal structure of the probability distribution functions, greatly boosting the accuracy on the target domain.

# Bibliography


[1] Xiong, X., De la Torre, F.: Supervised descent method and its applications to face alignment. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE (2013) 532–539

[2] Cao, X., Wei, Y., Wen, F., Sun, J.: Face alignment by explicit shape regression. International Journal of Computer Vision **107** (2014) 177–190

[3] Zhu, S., Li, C., Change Loy, C., Tang, X.: Face alignment by coarse-to-fine shape searching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 4998–5006

[4] Corneanu, C.A., Oliu, M., Cohn, J.F., Escalera, S.: Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications. Transactions on Pattern Analysis and Machine Intelligence Special Issue (2016)

[5] Li, S., Deng, W.: Deep facial expression recognition: A survey. IEEE transactions on affective computing (2020)

[6] Gao, M., Jiang, J., Zou, G., John, V., Liu, Z.: Rgb-d-based object recognition using multimodal convolutional neural networks: A survey. IEEE access **7** (2019) 43110–43136

[7] Serban, A., Poll, E., Visser, J.: Adversarial examples on object recognition: A comprehensive survey. ACM Computing Surveys (CSUR) **53** (2020) 1–38

[8] Chen, Y., Tian, Y., He, M.: Monocular human pose estimation: A survey of deep learning-based methods. Computer Vision and Image Understanding **192** (2020) 102897

[9] Aarthi, S., Chitrakala, S.: Scene understanding—a survey. In: 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), IEEE (2017) 1–4

[10] Nadeem, U., Shah, S.A.A., Sohel, F., Togneri, R., Bennamoun, M.: Deep learning for scene understanding. In: Handbook of deep learning applications. Springer (2019) 21–51

[11] Finn, C., Goodfellow, I., Levine, S.: Unsupervised learning for physical interaction through video prediction. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29, Curran Associates, Inc. (2016) 64–72

[12] Villegas, R., Yang, J., Hong, S., Lin, X., Lee, H.: Decomposing motion and content for natural video sequence prediction. In: 5th International Conference on Learning Representations. (2017)

[13] Denton, E.L., Birodkar, v.: Unsupervised learning of disentangled representations from video. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 4417–4426

[14] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1** (1989) 541–551

[15] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012) 1097–1105

[16] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Icml. (2010)

[17] Hochreiter, S.: Untersuchungen zu dynamischen neuronalen netzen. Diploma, Technische Universität München **91** (1991)

[18] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 1–9

[20] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778

[21] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9** (1997) 1735–1780

[22] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

[23] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)

[24] Xiong, X., De la Torre, F.: Global supervised descent method. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 2664–2673

[25] Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. arXiv preprint arXiv:1409.7495 (2014)

[26] Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Advances in neural information processing systems. (2016) 343–351

[27] Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3723–3732

[28] Lee, C.Y., Batra, T., Baig, M.H., Ulbricht, D.: Sliced wasserstein discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 10285–10295

[29] Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. IEEE Transactions on Pattern Analysis & Machine Intelligence (2001) 681–685

[30] Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active shape models-their training and application. Computer vision and image understanding **61** (1995) 38–59

[31] Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE (2013) 3476–3483

[32] Zhang, J., Shan, S., Kan, M., Chen, X.: Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. In: Computer Vision–ECCV 2014. Springer (2014) 1–16

[33] Burgos-Artizzu, X.P., Perona, P., Dollár, P.: Robust face landmark estimation under occlusion. In: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE (2013) 1513–1520

[34] Ren, S., Cao, X., Wei, Y., Sun, J.: Face alignment at 3000 fps via regressing local binary features. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 1685–1692

[35] Kazemi, V., Sullivan, J.: One millisecond face alignment with an ensemble of regression trees. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 1867–1874

[36] Jeni, L.A., Cohn, J.F., Kanade, T.: Dense 3d face alignment from 2d videos in real-time. In: Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on. Volume 1., IEEE (2015) 1–8

[37] Dollár, P., Welinder, P., Perona, P.: Cascaded pose regression. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE (2010) 1078–1085

[38] Lee, D., Park, H., Yoo, C.D.: Face alignment using cascade gaussian process regression trees. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE (2015) 4204–4212

[39] Alahi, A., Ortiz, R., Vandergheynst, P.: Freak: Fast retina keypoint. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, Ieee (2012) 510–517

[40] Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., Pantic, M.: 300 faces in-the-wild challenge: The first facial landmark localization challenge. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. (2013) 397–403

[41] Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2879–2886

[42] Belhumeur, P.N., Jacobs, D.W., Kriegman, D.J., Kumar, N.: Localizing parts of faces using a consensus of exemplars. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35** (2013) 2930–2940

[43] Le, V., Brandt, J., Lin, Z., Bourdev, L., Huang, T.S.: Interactive facial feature localization. In: Computer Vision–ECCV 2012. Springer (2012) 679–692

[44] Yin, L., Chen, X., Sun, Y., Worm, T., Reale, M.: A high-resolution 3d dynamic facial expression database. In: Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference On, IEEE (2008) 1–6

[45] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in neural information processing systems. (2014) 487–495

[46] Gross, R., Matthews, I., Cohn, J., Kanade, T., Baker, S.: Multi-pie. Image and Vision Computing **28** (2010) 807–813

[47] Zoumpourlis, G., Doumanoglou, A., Vretos, N., Daras, P.: Non-linear convolution filters for cnn-based learning. In: ICCV, IEEE (2017) 4771–4779

[48] Bergstra, J., Desjardins, G., Lamblin, P., Bengio, Y.: Quadratic polynomials learn better image features. Technical report, 1337 (2009)

[49] Livni, R., Shalev-Shwartz, S., Shamir, O.: On the computational efficiency of training neural networks. In: NIPS. (2014) 855–863

[50] Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In Bach, F., Blei, D., eds.: Proceedings of the 32nd International Conference on Machine Learning. Volume 37 of Proceedings of Machine Learning Research., PMLR (2015) 843–852

[51] Lotter, W., Kreiman, G., Cox, D.: Deep predictive coding networks for video prediction and unsupervised learning. In: International Conference on Learning Representations. (2016)

[52] Liu, Z., Yeh, R., Tang, X., Liu, Y., Agarwala, A.: Video frame synthesis using deep voxel flow. In: Proceedings of the International Conference on Computer Vision, IEEE, Curran Associates, Inc. (2017)

[53] Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S.: Action-Conditional Video Prediction using Deep Networks in Atari Games. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., eds.: Advances in Neural Information Processing Systems 28, Curran Associates, Inc. (2015) 2845–2853

[54] Ebert, F., Finn, C., Lee, A.X., Levine, S.: Self-supervised visual planning with temporal skip connections. arXiv preprint arXiv:1710.05268 (2017)

[55] SHI, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., WOO, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., eds.: Advances in Neural Information Processing Systems 28. Curran Associates, Inc. (2015) 802–810

[56] Patraucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. In: International Conference on Learning Representations Workshops. (2015)

[57] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)

[58] Michalski, V., Memisevic, R., Konda, K.: Modeling deep temporal dependencies with recurrent grammar cells. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 27. Curran Associates, Inc. (2014) 1925–1933

[59] Cricri, F., Honkala, M., Ni, X., Aksu, E., Gabbouj, M.: Video ladder networks. arXiv preprint arXiv:1612.01756 (2016)

[60] Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., Kavukcuoglu, K.: Video pixel networks. In Precup, D., Teh, Y.W., eds.: Proceedings of the 34th International Conference on Machine Learning. Volume 70 of Proceedings of Machine Learning Research., PMLR (2017) 1771–1779

[61] Liang, X., Lee, L., Dai, W., Xing, E.P.: Dual motion gan for future-flow embedded video prediction. In: Proceedings of the International Conference on Computer Vision, IEEE, Curran Associates, Inc. (2017) 1762–1770

[62] Vukotić, V., Pintea, S.L., Raymond, C., Gravier, G., Van Gemert, J.: One-step time-dependent future video frame prediction with a convolutional encoder-decoder neural network. In: Netherlands Conference on Computer Vision. (2017)

[63] Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R.H., Levine, S.: Stochastic variational video prediction. In: 6th International Conference on Learning Representations. (2018)

[64] Prémont-Schwarz, I., Ilin, A., Hao, T., Rasmus, A., Boney, R., Valpola, H.: Recurrent ladder networks. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30, Curran Associates, Inc. (2017) 6009–6019

[65] Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29, Curran Associates, Inc. (2016) 667–675

[66] Xue, T., Wu, J., Bouman, K., Freeman, B.: Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29, Curran Associates, Inc. (2016) 91–99

[67] Van Amersfoort, J., Kannan, A., Ranzato, M., Szlam, A., Tran, D., Chintala, S.: Transformation-based models of video sequences. arXiv preprint arXiv:1701.08435 (2017)

[68] Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: Proceedings of the International Conference on Computer Vision, IEEE, Curran Associates, Inc. (2017) 3332–3341

[69] Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. In: International Conference on Learning Representations (ICLR). (2016)

[70] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K., eds.: Advances in Neural Information Processing Systems 27. Curran Associates, Inc. (2014) 2672–2680

[71] Sedaghat, N., Zolfaghari, M., Brox, T.: Hybrid learning of optical flow and next frame prediction to boost optical flow in the wild. arXiv preprint arXiv:1612.03777 (2016)

[72] Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In Kittler, J., Petrou, M., Nixon, M.S., eds.: Proceedings of the 17th International Conference on Pattern Recognition. Volume 3., IEEE (2004) 32–36

[73] Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)

[74] Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 30. (2016)

[75] Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: European conference on computer vision, Springer (2016) 443–450

[76] Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. arXiv preprint arXiv:1502.02791 (2015)

[77] Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., Sriperumbudur, B.K.: Optimal kernel choice for large-scale two-sample tests. In: Advances in neural information processing systems, Citeseer (2012) 1205–1213

[78] Xu, M., Zhang, J., Ni, B., Li, T., Wang, C., Tian, Q., Zhang, W.: Adversarial domain adaptation with domain mixup. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 34. (2020) 6502–6509

[79] Li, J., Chen, E., Ding, Z., Zhu, L., Lu, K., Shen, H.T.: Maximum density divergence for domain adaptation. IEEE transactions on pattern analysis and machine intelligence (2020)