Efficient Modeling of High Order Functions in Computer Vision

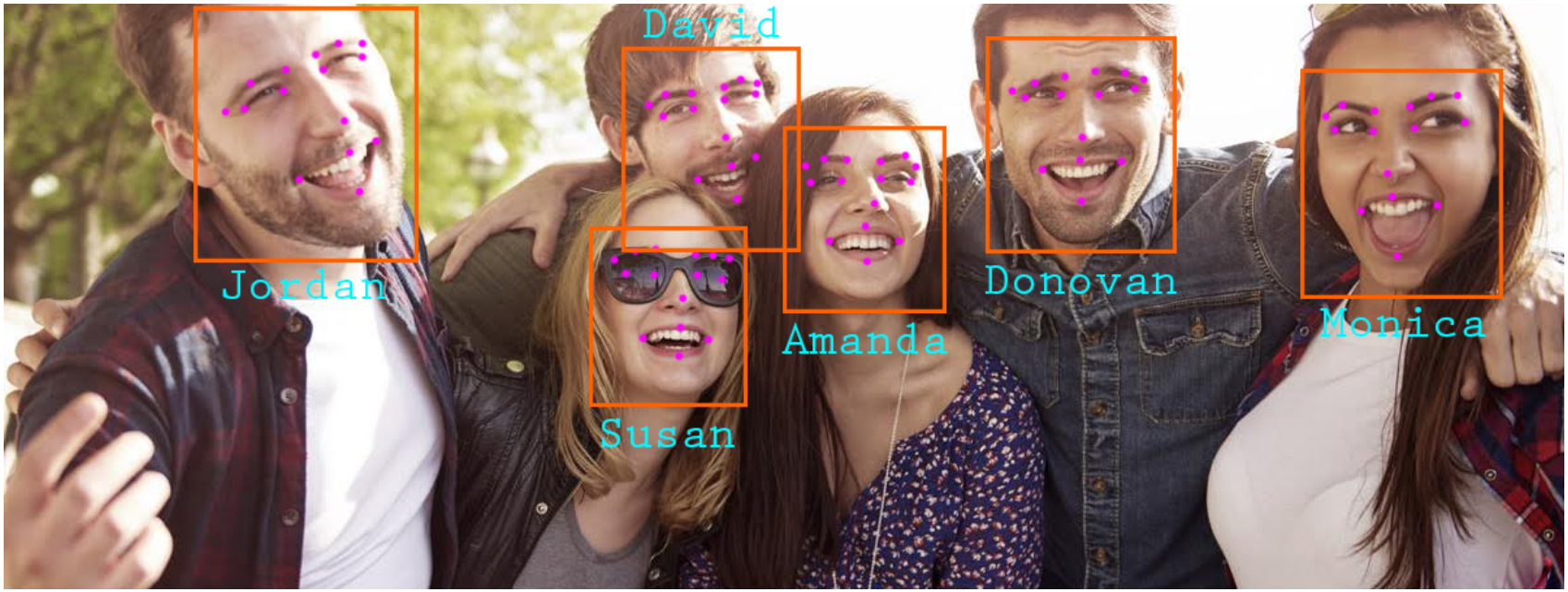
Marc Oliu Simón

supervised by

Dr. Sergio Escalera Guerrero
Dr. Xavier Baró Solé

Efficiency in Neural Networks

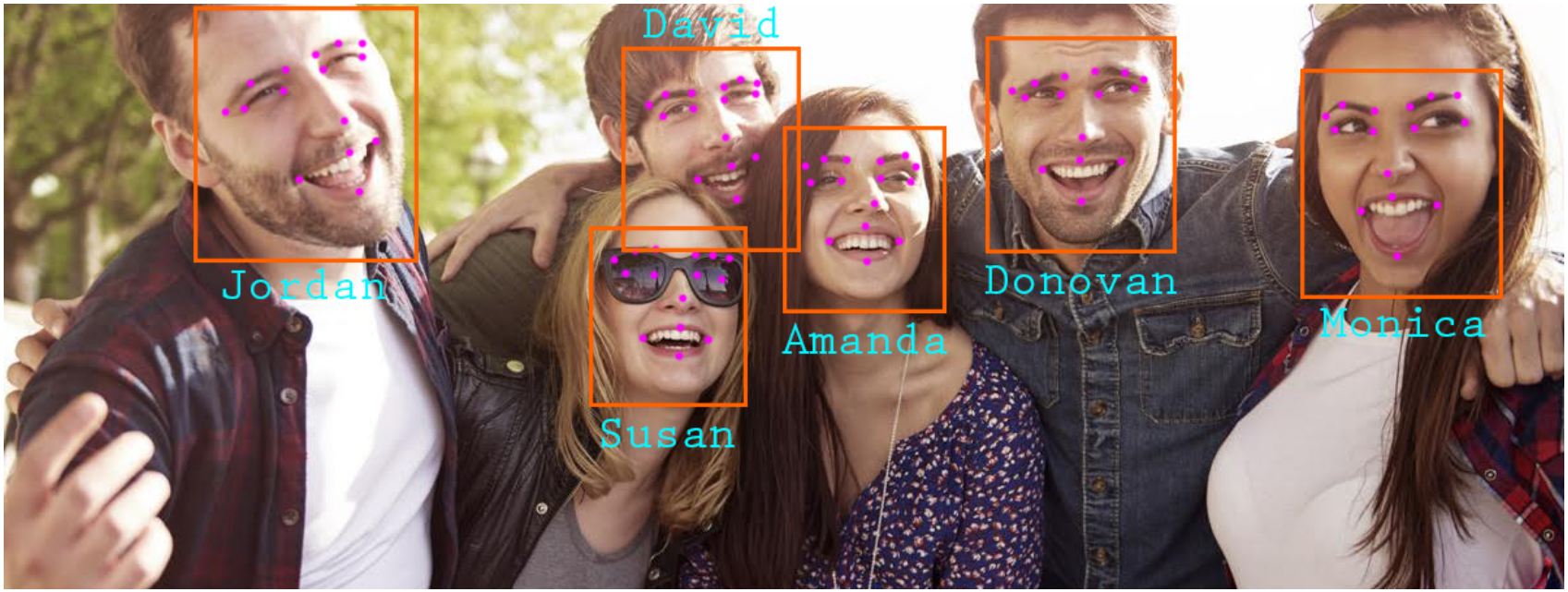
Neural Networks and GPUs have brought great progress to Computer Vision, with many previously unsolvable tasks being solved in a short few years.



Face Detection

Efficiency in Neural Networks

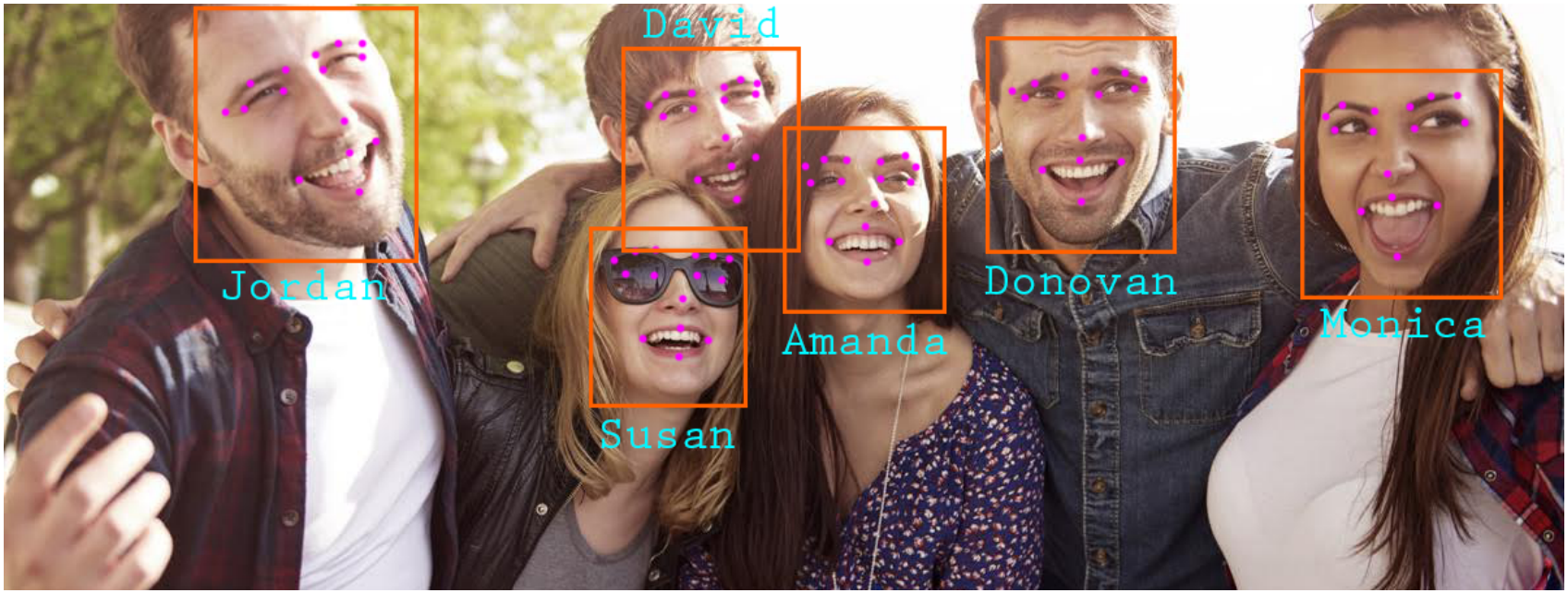
Neural Networks and GPUs have brought great progress to Computer Vision, with many previously unsolvable tasks being solved in a short few years.



Facial Landmark Localization

Efficiency in Neural Networks

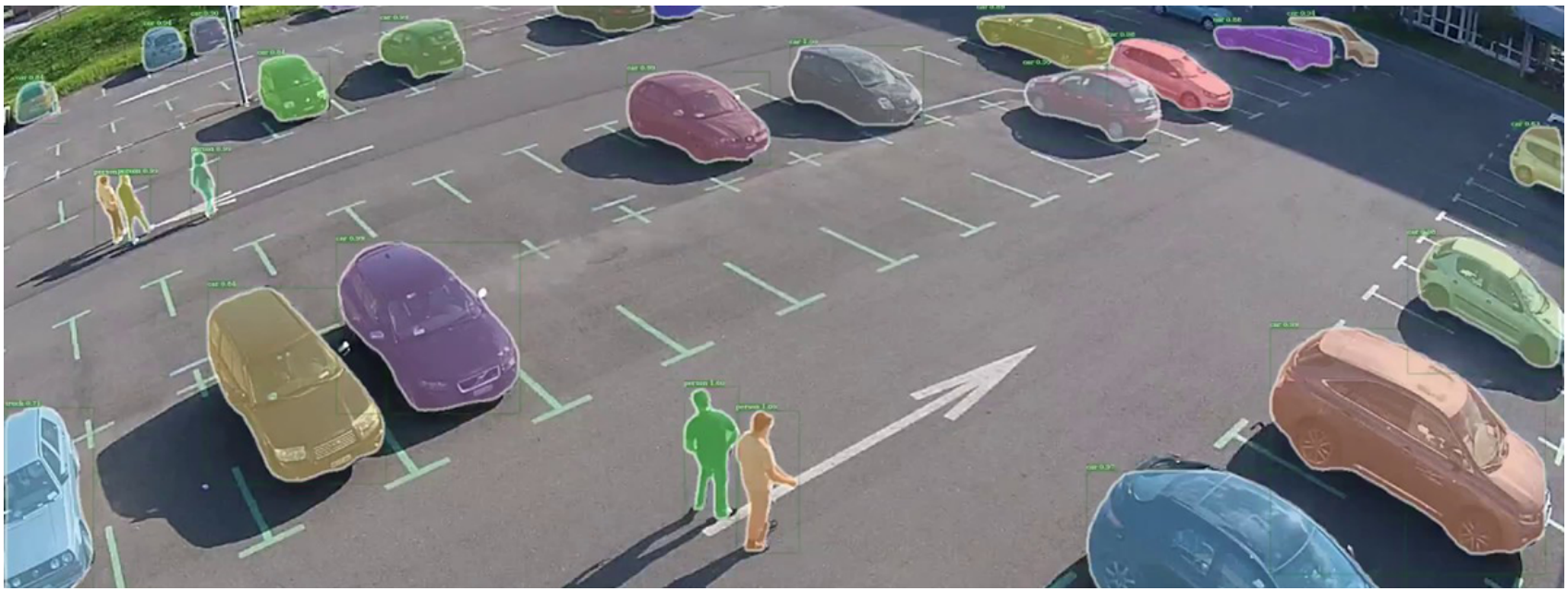
Neural Networks and GPUs have brought great progress to Computer Vision, with many previously unsolvable tasks being solved in a short few years.



Face Recognition

Efficiency in Neural Networks

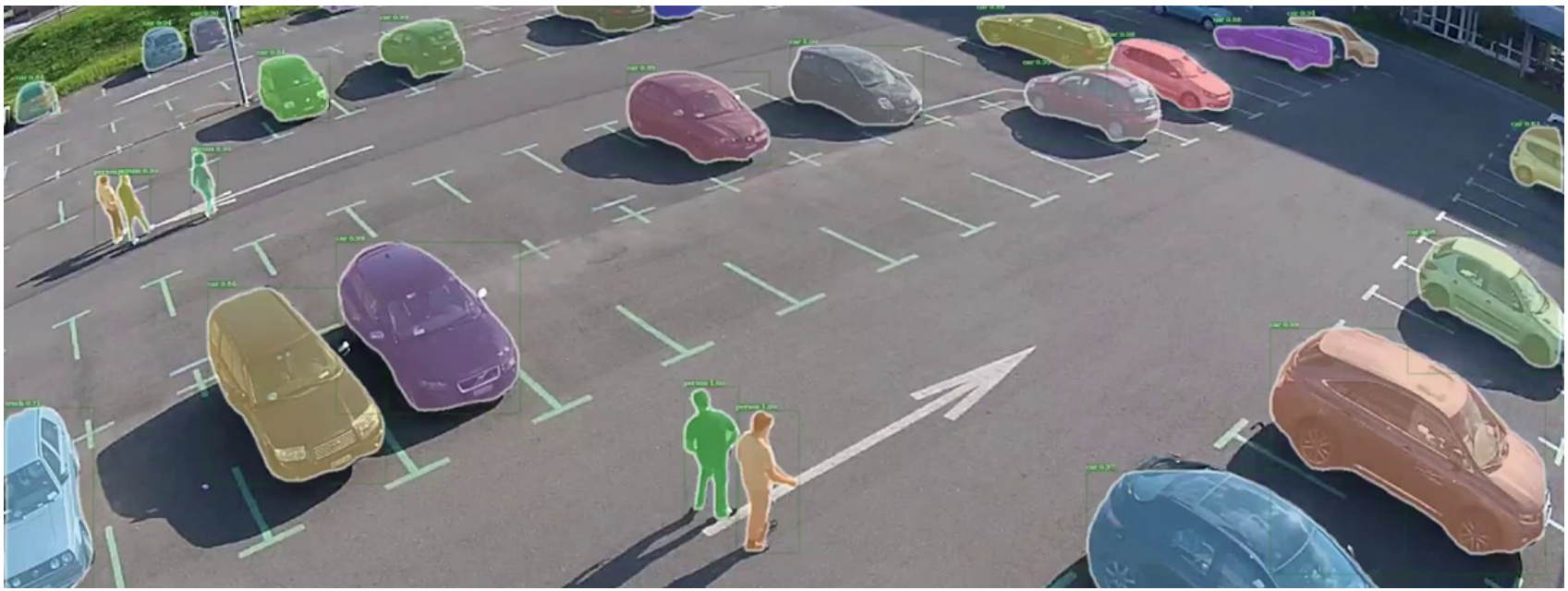
Neural Networks and GPUs have brought great progress to Computer Vision, with many previously unsolvable tasks being solved in a short few years.



Object Detection

Efficiency in Neural Networks

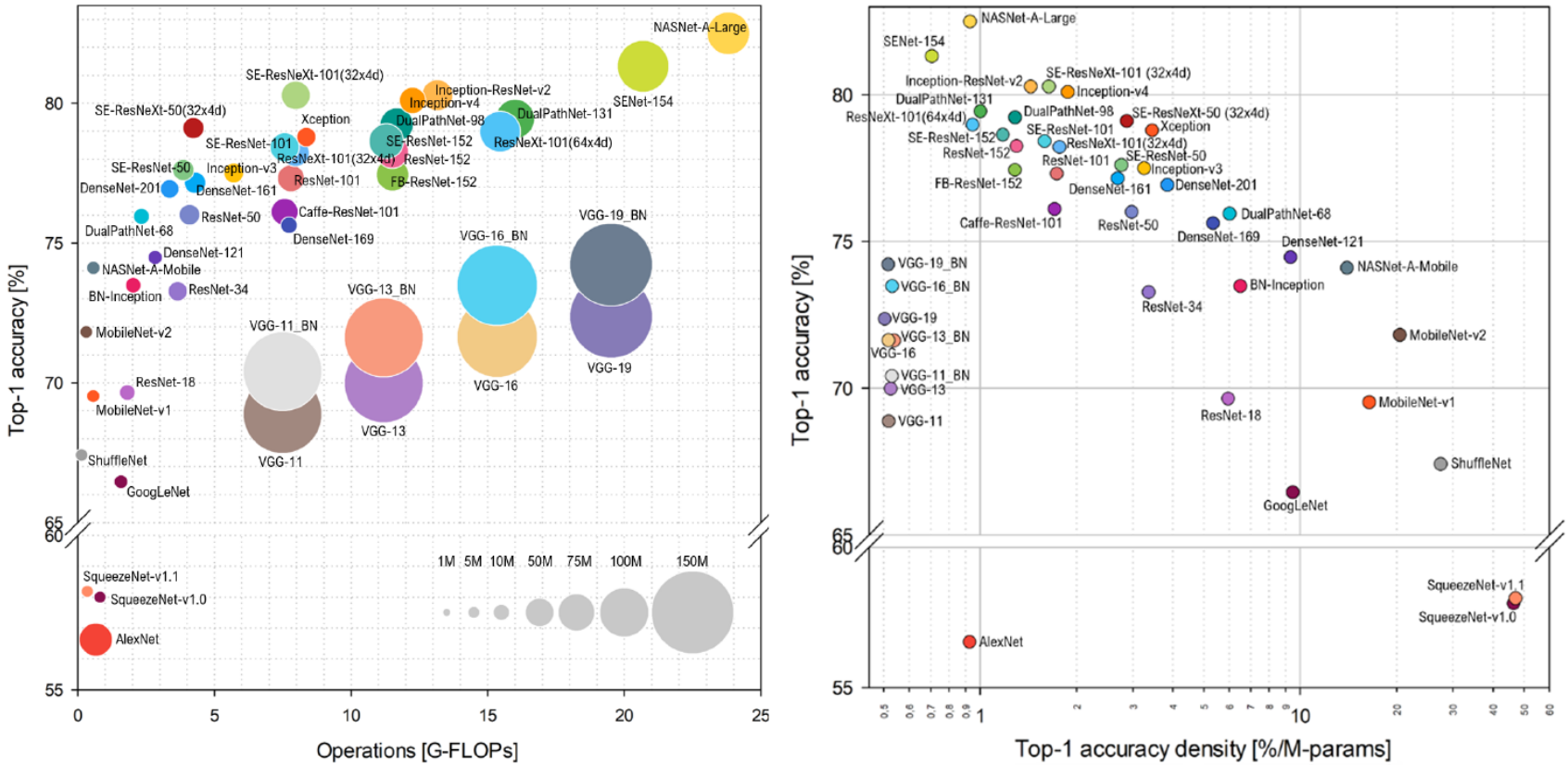
Neural Networks and GPUs have brought great progress to Computer Vision, with many previously unsolvable tasks being solved in a short few years.



Object Segmentation

Efficiency in Neural Networks

Computational and memory costs

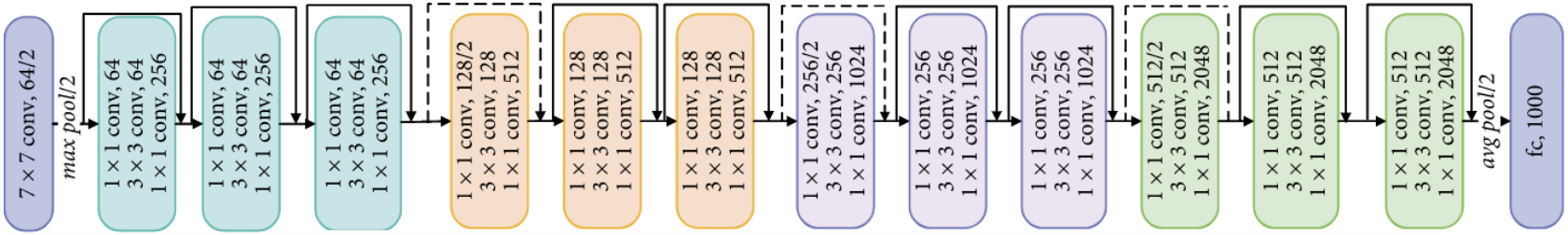


Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). *Benchmark analysis of representative deep neural network architectures.* IEEE Access (64270-64277).

Efficiency in Neural Networks

A good example: ResNet

An example of efficient modeling is the popular ResNet topology.

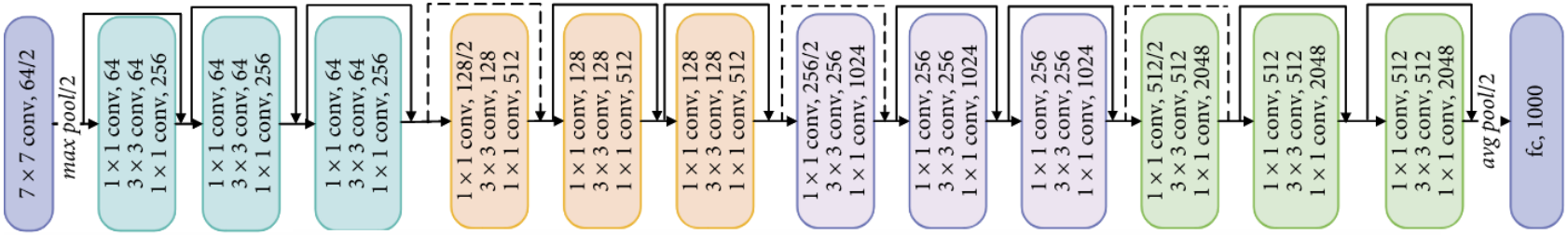


- Supports very deep topologies
- Provides low-order shortcuts

Efficiency in Neural Networks

A good example: ResNet

An example of efficient modeling is the popular ResNet topology.



- Supports very deep topologies
- Provides low-order shortcuts

High order methods

Definition of order

We define the order of a model as the highest degree polynomial among the polynomial expansion of its dependent variables relative to the independent input variables.

$$\arg\min_{W} ||XW - Y||_2^2$$

A **first order** linear regressor directly combines the independent input variables X in order to predict the dependent variables Y.

High order methods

Definition of order

We define the order of a model as the highest degree polynomial among the polynomial expansion of its dependent variables relative to the independent input variables.

$$\arg\min_{W} ||(X \odot X)W - Y||_2^2$$

A **second order** linear regressor, also known as polynomial regressor, also considers pairs of independent variables as inputs.

High order methods

Order in neural networks

In strict terms, neural networks do not possess an order relationship. This is due to the non-linear activation functions commonly used.

$$y = \sigma(Wx + b)$$

That said, we may consider the case where we use $\sigma(x) = x^2$ as our non-linear activation function and extrapolate from there.

Order in neural networks

Quadratic neural networks

Quadratic models have been proposed as replacements to fully connected layers in order to increase the expressiveness of neural networks. A common approach is using a Volterra series to describe a 2nd order multiple regression model.

$$y_i = x^T W_2 x + x^T W_1 + b$$

Zoumpourlis, G., Doumanoglou, A., Vretos, N. & Daras, P. (2017). *Non-linear convolution filters for cnn-based learning.* In ICCV (4771-4779).

Order in neural networks

Quadratic neural networks

We first find a more compact notation for the quadratic layer.

$$y_i(x) = \tilde{x}^T W \tilde{x}$$

$$W = \begin{bmatrix} (W_2)_{1,1} & (W_2)_{1,2} & .. & (W_1)_1 \\ (W_2)_{2,1} & (W_2)_{2,2} & .. & (W_1)_2 \\ .. & .. & .. & .. \\ 0 & 0 & .. & b \end{bmatrix}$$

Here, $\tilde{x} = [x; 1]$. Please note that $\tilde{x}^T W \tilde{x} = \tilde{x}^T W^T \tilde{x}$. Thus there must always exist a symmetric W for any given W_1, W_2 and b.

Order in neural networks

Equivalence to polynomial networks

Using eigendecomposition on W we obtain the following formula:

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

Given $\tilde{M}$ a selector that gathers the basis defining the quadratic matrix of each neuron, we can now find an expression for all neurons y_i in the layer:

$$y(X) = \left[(X\overline{U}) \circ (X\overline{U}) \right] M$$

$$\textit{where } \overline{U} = \left[U^{(1)}, \ ..., \ U^{(n)} \right], M = \begin{bmatrix} S^{(1)} & 0 & 0 \\ 0 & .. & 0 \\ 0 & 0 & S^{(n)} \end{bmatrix} \widetilde{M}$$

Order in neural networks

Equivalence to polynomial networks

By defining $g(X) = X^{\circ 2}$ as an activation function, we obtain the following equation, which corresponds to a polynomial network.

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

- Two-layer model with quadratic activation
- Polynomial networks are more compact

Livni, R., Shalev-Shwartz, S. & Shamir, O. (2014). *On the computational efficiency of training neural networks.* In NIPS (855-863).

Order in neural networks

Equivalence to polynomial networks

By defining $g(X) = X^{\circ 2}$ as an activation function, we obtain the following equation, which corresponds to a polynomial network.

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

- Two-layer model with quadratic activation
- Polynomial networks are more compact

Livni, R., Shalev-Shwartz, S. & Shamir, O. (2014). *On the computational efficiency of training neural networks.* In NIPS (855-863).

Order in neural networks

Equivalence to polynomial networks

By defining $g(X) = X^{\circ 2}$ as an activation function, we obtain the following equation, which corresponds to a polynomial network.

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

- Two-layer model with quadratic activation
- Polynomial networks are more compact

Livni, R., Shalev-Shwartz, S. & Shamir, O. (2014). *On the computational efficiency of training neural networks.* In NIPS (855-863).

Order in neural networks

Equivalence to neural networks

Nesting two quadratic layers based on the compact form we found, we obtain the following form:

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

Both $M^{(1)}$ and $\overline{U}^{(2)}$ are trainable parameters. Thus we can simplify the following equation:

$$y(X) = \left(\left[\left(X\overline{U}^{(1)} \right)^{\circ 2}, \ \mathbf{1}^{<q \times 1>} \right] \tilde{U}^{(2)} \right)^{\circ 2} M^{(2)}$$

Order in neural networks

Equivalence to neural networks

The resulting equation corresponds to a three-layer model with quadratic activation functions and a linear output layer.

$$y(X) = \left(\left[\left(X \overline{U}^{(1)} \right)^{\circ 2}, \; \mathbf{1}^{<q \times 1>} \right] \tilde{U}^{(2)} \right)^{\circ 2} M^{(2)}$$

- A neural network with l layers and quadratic activation functions is equivalent to a 2^{l-1}-th order regressor.
- Other activations result in a model different from an n-th order regressor that captures a similar level of complexity.
- Neural networks are more compact than regressors of the same order due to the sharing of basis.

Order in neural networks

Equivalence to neural networks

The resulting equation corresponds to a three-layer model with quadratic activation functions and a linear output layer.

$$y(X) = \left(\left[\left(X \overline{U}^{(1)} \right)^{\circ 2}, \; \mathbf{1}^{<q \times 1>} \right] \tilde{U}^{(2)} \right)^{\circ 2} M^{(2)}$$

- A neural network with l layers and quadratic activation functions is equivalent to a 2^{l-1}-th order regressor.
- Other activations result in a model different from an n-th order regressor that captures a similar level of complexity.
- Neural networks are more compact than regressors of the same order due to the sharing of basis.

Order in neural networks

Equivalence to neural networks

The resulting equation corresponds to a three-layer model with quadratic activation functions and a linear output layer.

$$y(X) = \left(\left[\left(X \overline{U}^{(1)} \right)^{\circ 2}, \; \mathbf{1}^{<q \times 1>} \right] \tilde{U}^{(2)} \right)^{\circ 2} M^{(2)}$$

- A neural network with l layers and quadratic activation functions is equivalent to a 2^{l-1}-th order regressor.
- Other activations result in a model different from an n-th order regressor that captures a similar level of complexity.
- Neural networks are more compact than regressors of the same order due to the sharing of basis.

Order in neural networks

Equivalence to neural networks

The resulting equation corresponds to a three-layer model with quadratic activation functions and a linear output layer.

$$y(X) = \left(\left[\left(X \overline{U}^{(1)} \right)^{\circ 2}, \; \mathbf{1}^{<q \times 1>} \right] \tilde{U}^{(2)} \right)^{\circ 2} M^{(2)}$$

- A neural network with l layers and quadratic activation functions is equivalent to a 2^{l-1}-th order regressor.
- Other activations result in a model different from an n-th order regressor that captures a similar level of complexity.
- Neural networks are more compact than regressors of the same order due to the sharing of basis.

Efficient modeling of high order functions

Being clever when defining the model

We will see **three different approaches** to efficiently modeling high order functions, each associated to a specific Computer Vision task.

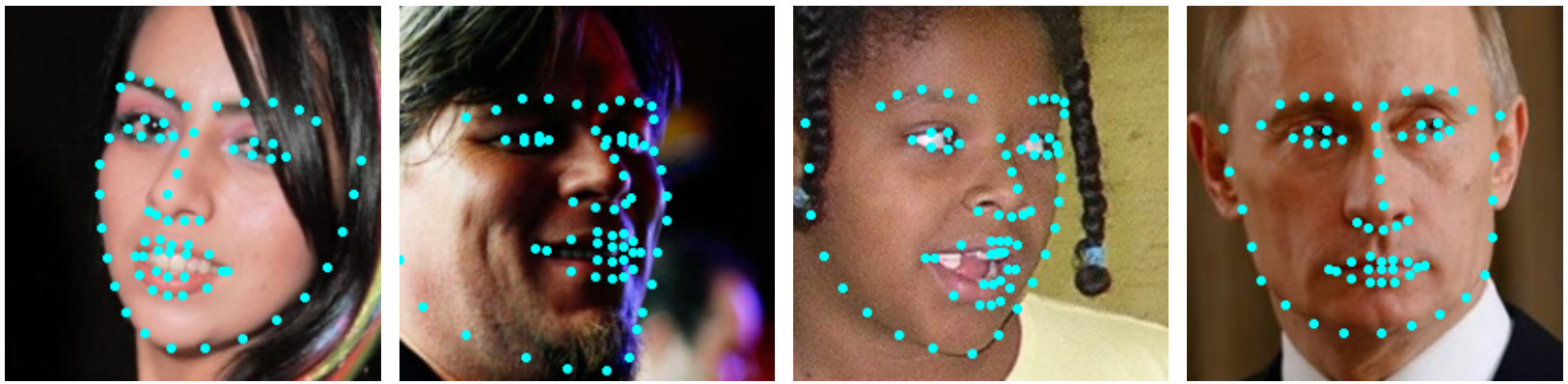
- Limiting variable interactions
- Providing low-order shortcuts
- Stochastic approximation of complex functions

Second Order Linear Methods

Continuous Supervised Descent Method

Facial Landmark Localization

Problem definition



Facial landmark localization (aka. face alignment) is a processing step common to many face analysis techniques. It locates a series of points of interest in a face image.

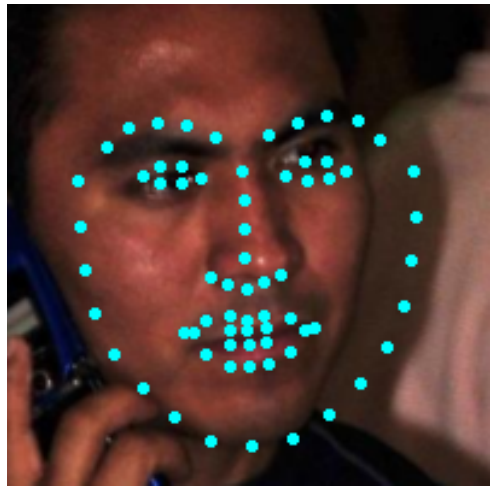
- Problem partially solved for near-frontal faces
- Some difficulties for extreme shadows and rotations
- The more robust approaches are expensive to train

Facial Landmark Localization

Cascaded regression

Usually solved by sequentially applying a series of regression functions f^i mapping the features Φ^i, extracted using the current shape estimate X^i, to the difference between the estimate and ground truth shapes $\Delta X^i = X^i - X^*$.

$$\begin{aligned} X^{i+1} &= X^i + \Delta X^i \\ &= X^i + f^i(\Phi^i) \end{aligned}$$



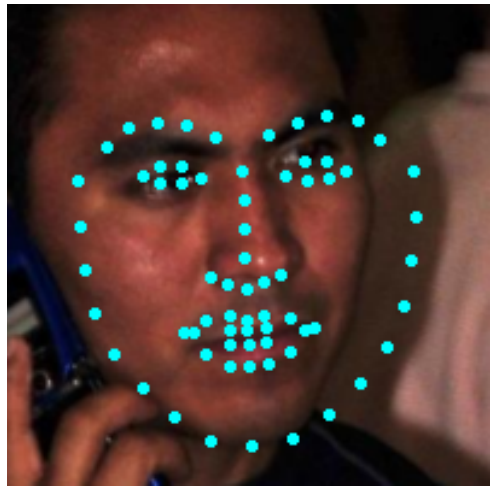
Xiong, X., & De la Torre, F. (2013). *Supervised descent method and its applications to face alignment.* In CVPR (pp. 532-539).

Facial Landmark Localization

Cascaded regression

Usually solved by sequentially applying a series of regression functions f^i mapping the features Φ^i, extracted using the current shape estimate X^i, to the difference between the estimate and ground truth shapes $\Delta X^i = X^i - X^*$.

$$
\begin{aligned}
X^{i+1} &= X^i + \Delta X^i \\
&= X^i + f^i(\Phi^i)
\end{aligned}
$$



Xiong, X., & De la Torre, F. (2013). *Supervised descent method and its applications to face alignment.* In CVPR (pp. 532-539).

Facial Landmark Localization

Global Supervised Descent Method

Suppose an ideal function $\Delta X^i = f(\Phi)$ mapping the features Φ to targets ΔX^i. We can express it with as $\Delta X^i = \Phi^i W^i$, where $W^i = g(\Phi^i)$. Can we approximate the weights space?

GSDM solution: Partition the space into quadrants across a projected feature subspace $\widetilde{\Phi}^i = \Phi^i P$. Learn a linear regressor for each quadrant.

Xiong, X. & De la Torre, F. (2015). *Global supervised descent method.* In CVPR (2664-2673).

Facial Landmark Localization

Global Supervised Descent Method

Advantages

- Adds robustness to the features main modes of variation
- Approximate $g(\Phi^i)$ non-linearly

Disadvantages

- Low granularity approximating $g(\Phi^i)$
- Number of weights grows exponentially wrt. $||\widetilde{\Phi}^i||$
- Logarithmic reduction on number of samples contributing to each weight

Continuous Supervised Descent Method

Space of linear regressors

CSDM Solution: Define a linear regressor approximating $g(\Phi^i)$ given the feature subspace $\widetilde{\Phi}^i$.

This corresponds to a second order polynomial regression where the projection matrix P restricts the combination of variables in Φ^i.

$$\arg\min_{R_j^i} ||(\Phi^i \circ (\widetilde{\Phi^i} R_j^i))\mathbb{1}_{(k+1)} - \Delta X_j^i||_2^2$$

Continuous Supervised Descent Method

Space of linear regressors

CSDM Solution: Define a linear regressor approximating $g(\Phi^i)$ given the feature subspace $\widetilde{\Phi}^i$.

Which can be expressed as a linear regression problem by expanding the features using the Khatri-Rao product.

$$\arg\min_{R_j^i} ||(\widetilde{\Phi^i} \odot \Phi^i) vec(R_j^{i\mathsf{T}}) - \Delta X_j^i||_2^2$$

Continuous Supervised Descent Method

Advantages and disadvantages

Compared to the method most similar to ours, Global SDM, our approach has the following pros and cons.

Advantages

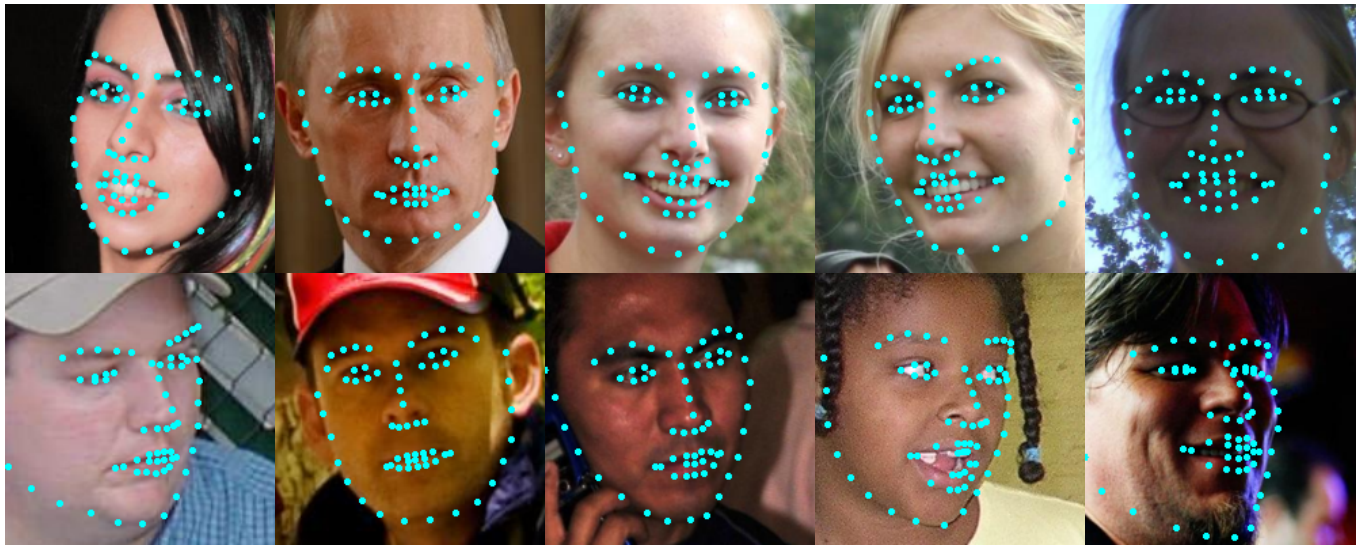
- Adds robustness to the features main modes of variation
- Continuous approximation of $g(\Phi^i)$
- Linear growth in number of parameters wrt. $||\widetilde{\Phi}^i||$
- All instances contribute to each parameter

Disadvantages

- Approximate $g(\Phi^i)$ linearly

Datasets

300-W



- 3148 train and 689 test samples
- 68 facial landmarks
- No extreme face poses

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2013). *300 faces in-the-wild challenge: The first facial landmark localization challenge.* ICCV Workshop (397-403).

Datasets

Proposed: BU4DFE-Synthetic

- 75k images, synthetically rotated from BU4DFE
- Rotations between $\pm 90^\circ$ in yaw and $\pm 45^\circ$ in pitch
- Backgrounds sampled from the Places-205 test set

Yin, L., Chen, X., Sun, Y., Worm, T., & Reale, M. (2008). *A high-resolution 3D dynamic facial expression database.* FG (1-6).

Quantitative results

Comparison to the state of the art

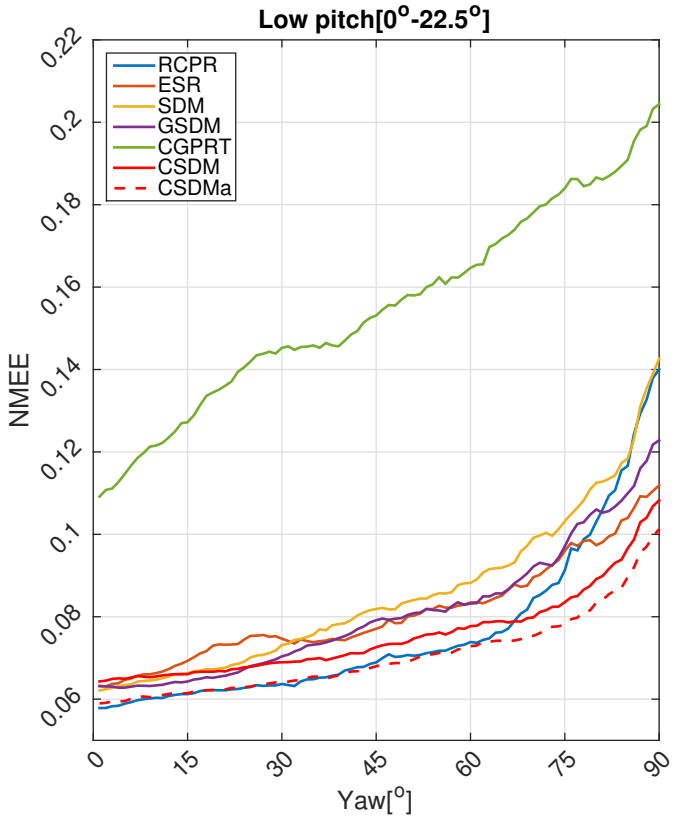
$$NMEE = \frac{1}{n} \frac{\sum_i ||x_i - x_i^*||_2}{||x_l^* - x_r^*||_2}$$

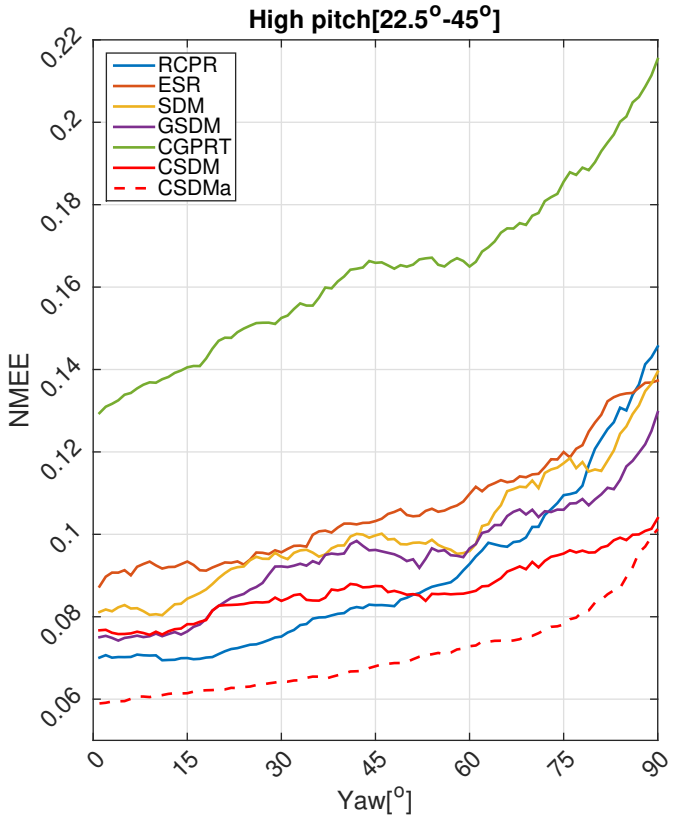
	ESR	RCPR	SDM	ERT	LBF	CGPRT	CFSS	GSDM	CSDM	CSDMa
300W	7.58	8.38	7.52	6.40	6.32	**5.71**	5.76	6.96	6.83	6.40
BU4DFE-S	9.45	8.61	9.57	-	-	15.81	-	9.01	8.28	**7.62**

Table: Comparison with state-of-the-art methods NMEE without (CSDM) and with multiple test initialisations (CSDMa).

Quantitative results

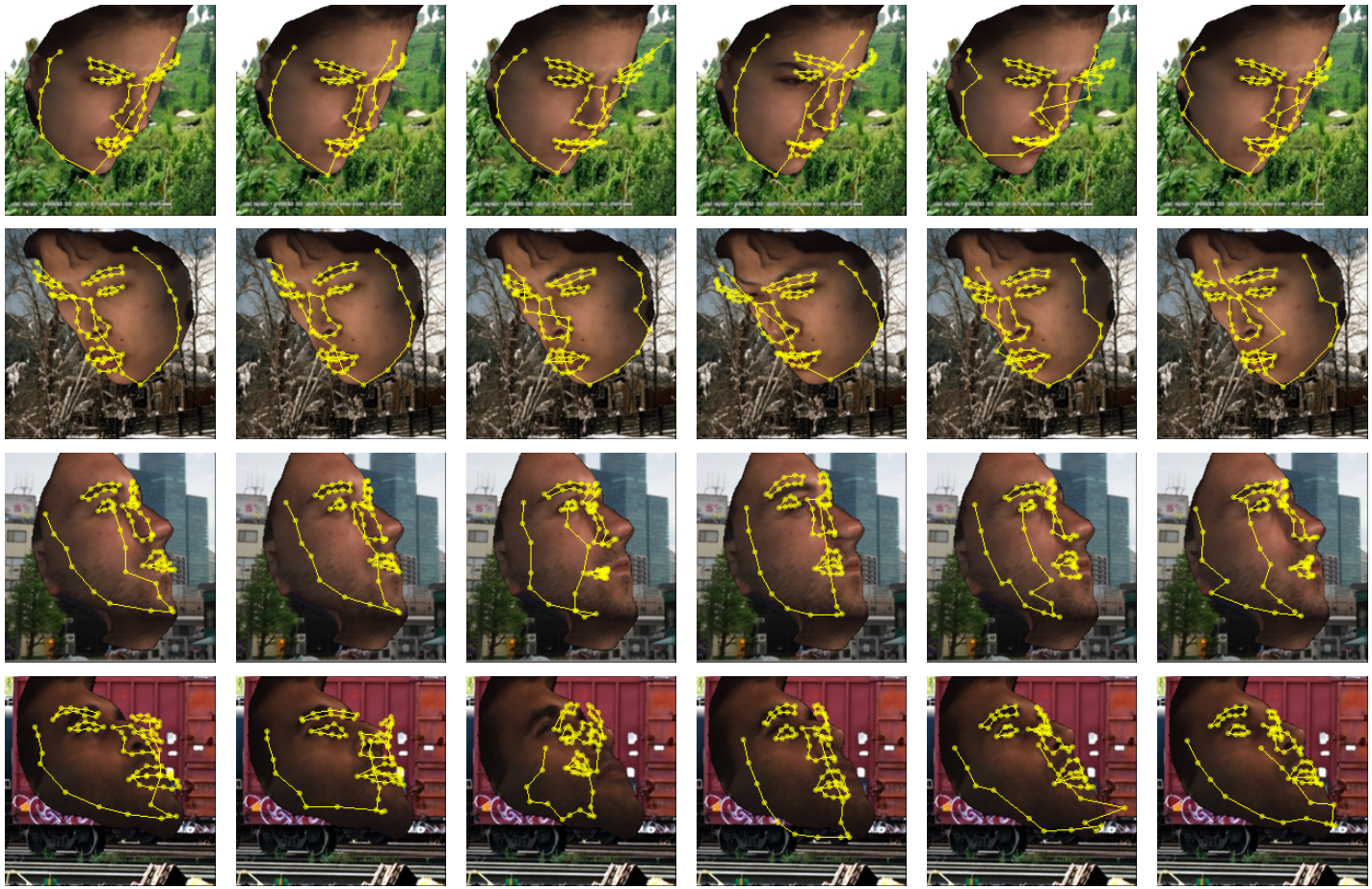
Robustness to pose on BU4DFE-S





Qualitative results

Test samples using different approaches



Conclusions

Contributions

- Natural generalisation of SDM
- Continuous, more adaptive approach to regressor selection

Strengths

- Highly robust to the head pose
- Smaller memory footprint
- Reduced need for training instances

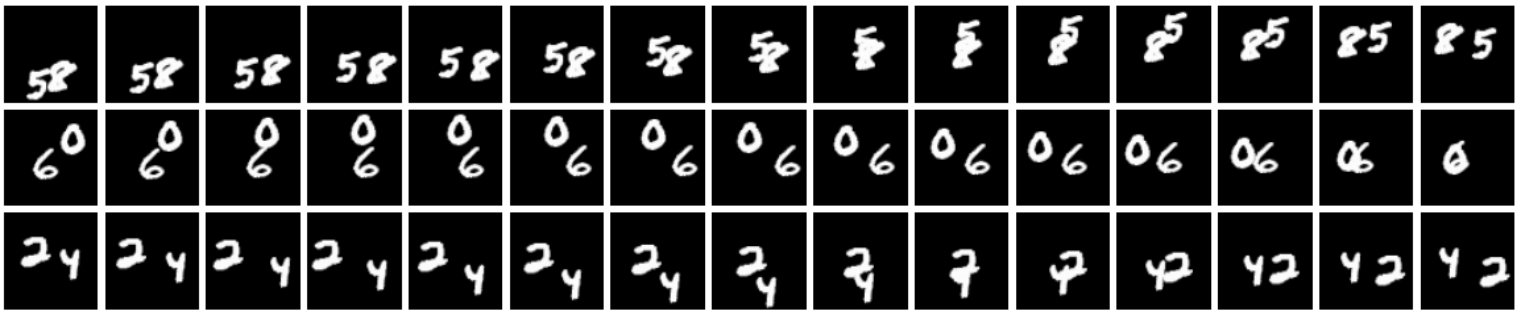
Shared-state Neural Networks

Folded Recurrent Neural Networks

Future Video Prediction

Generate the following frames given a video sequence

Given K initial frames, predict the following N frames. Ideally, we want to predict them without feedback from the ground truth. As we go further away from the last input frame, the problem is becomes harder.

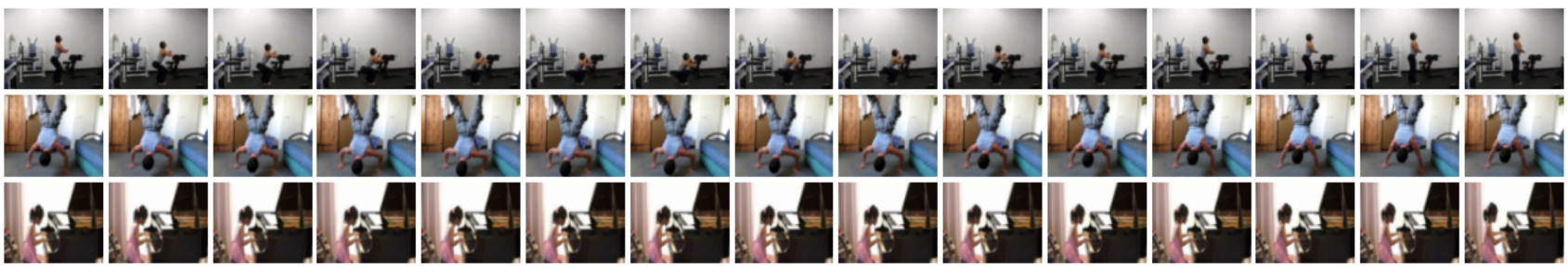


- Relatively easy for unimodal futures
- More complicated for multiple possible futures

Future Video Prediction

Generate the following frames given a video sequence

Given K initial frames, predict the following N frames. Ideally, we want to predict them without feedback from the ground truth. As we go further away from the last input frame, the problem is becomes harder.

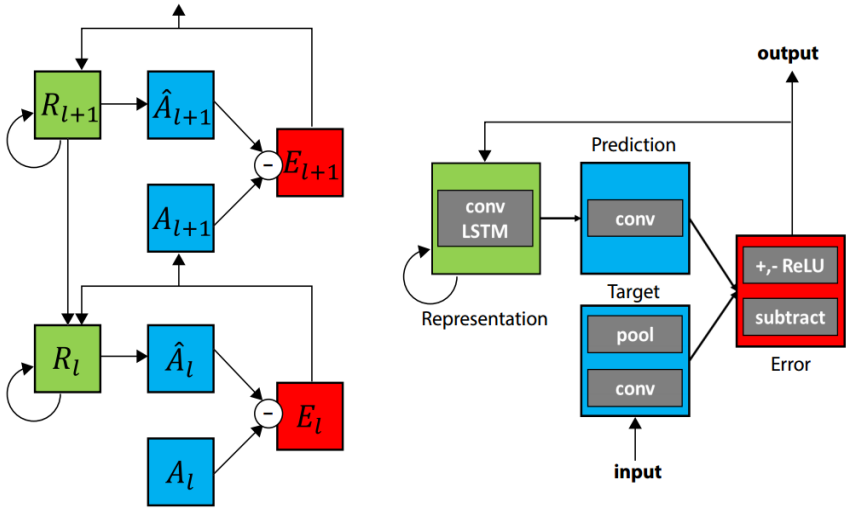


- Relatively easy for unimodal futures
- More complicated for multiple possible futures

Future Video Prediction

Commonly solved trough recurrent convolutional AEs

Recurrent convolutional autoencoders can encode both spatial information and the temporal dynamics of the sequence.

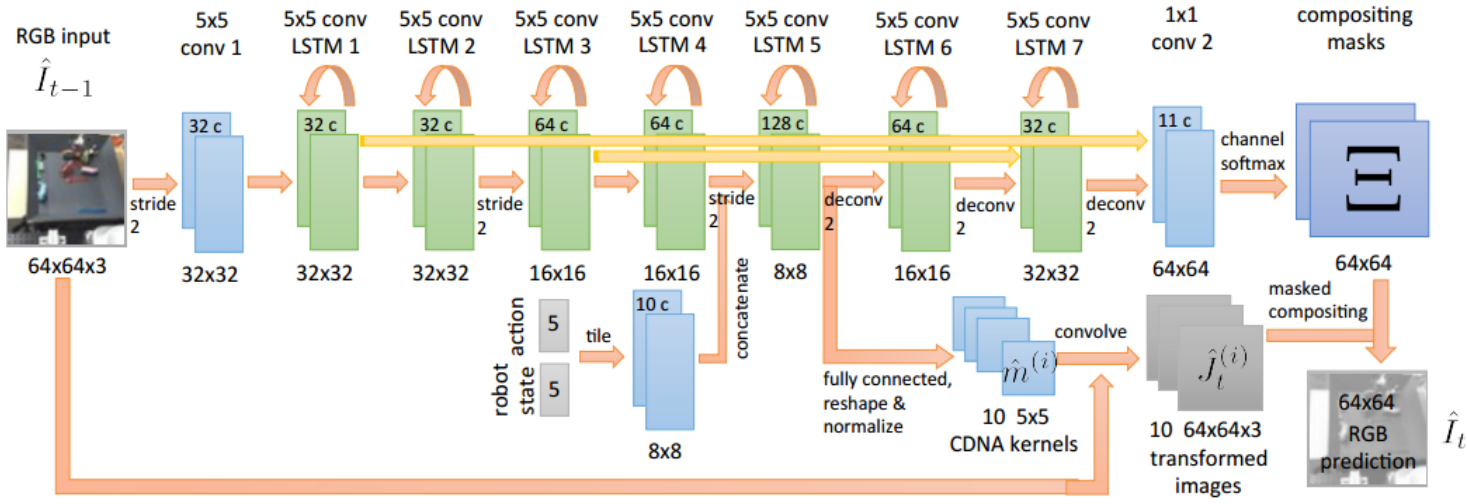


Lotter, W., Kreiman, G., & Cox, D. (2016). *Deep predictive coding networks for video prediction and unsupervised learning.* arXiv preprint arXiv:1605.08104.

Future Video Prediction

Commonly solved trough recurrent convolutional AEs

Recurrent convolutional autoencoders can encode both spatial information and the temporal dynamics of the sequence.

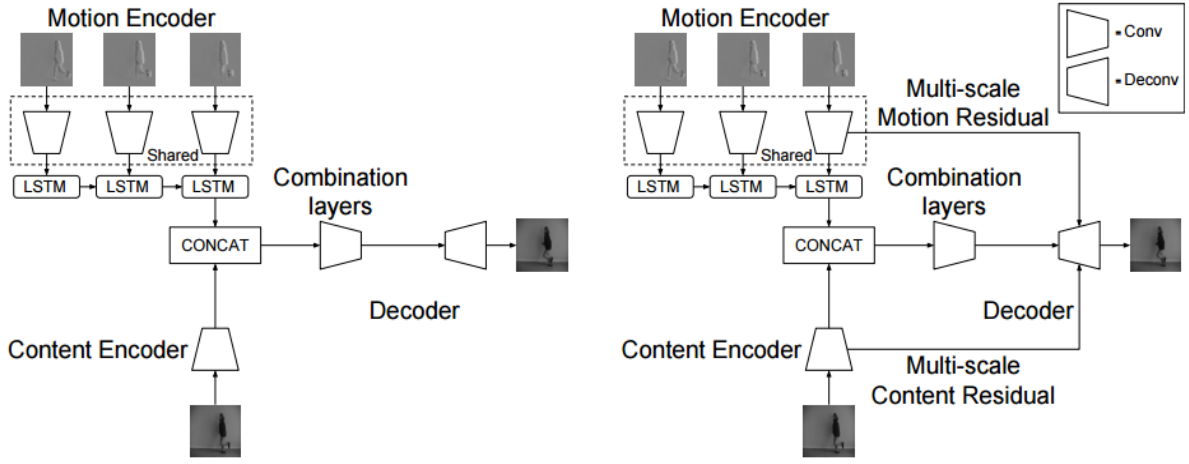


Finn, C., Goodfellow, I., & Levine, S. (2016). *Unsupervised learning for physical interaction through video prediction.* In NIPS (pp. 64-72).

Future Video Prediction

Commonly solved trough recurrent convolutional AEs

Recurrent convolutional autoencoders can encode both spatial information and the temporal dynamics of the sequence.



(a) Base MCnet

(b) MCnet with Multi-scale Motion-Content Residuals

Villegas, R., Yang, J., Hong, S., Lin, X., & Lee, H. (2017). *Decomposing motion and content for natural video sequence prediction.* arXiv preprint arXiv:1706.08033.

Future Video Prediction

Problems and limitations

- Multiple possible futures
- Blur/error propagation through time
- Network capacity contraints

For sequences with **multiple possible futures**, a standard gecurrent auto-encoder will average these possible futures into a single blurry prediction.

Future Video Prediction

Problems and limitations

- Multiple possible futures
- Blur/error propagation through time
- Network capacity contraints

Blur and errors introduced in one prediction **are propagated through time**: These errors are fed back into the network to predict the following frame.

Future Video Prediction

Problems and limitations

- Multiple possible futures
- Blur/error propagation through time
- Network capacity contraints

The dynamics of the whole frame must be captured and projected to the future. This is done recurrently for each time step. The **memory requirements increase rapidly**.

Folded Recurrent Neural Networks

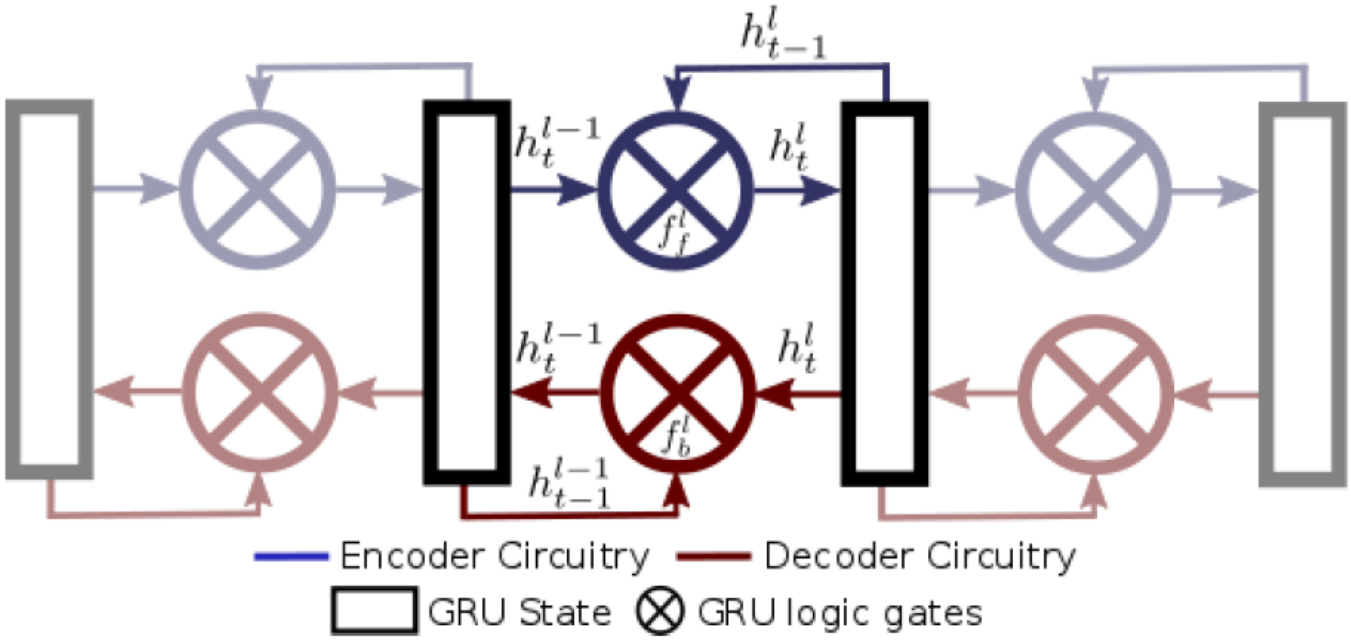
Proposal: Outline

We want to limit the propagation of blur and errors by doing away with the prediction feedback. To do so, we propose folding the network in half, sharing the states between the encoder and decoder.

- The states can be updated bidirectionally (pipeline reversal)
- Only the encoder/decoder is used at each time step
- **Encoder:** Updates the dynamics with new input frames
- **Decoder:** Projects the dynamics into the future

Folded Recurrent Neural Networks

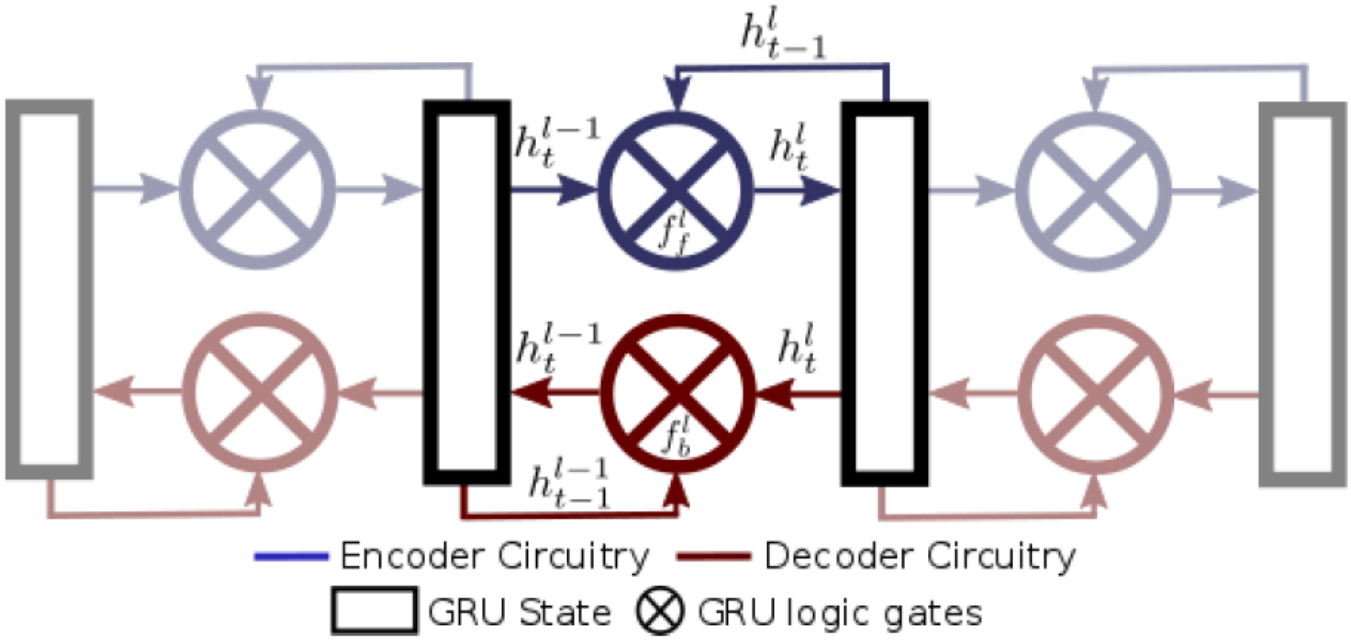
Proposal: Bijective GRU (bGRU)



Regular GRUs fully expose their state as output

Folded Recurrent Neural Networks

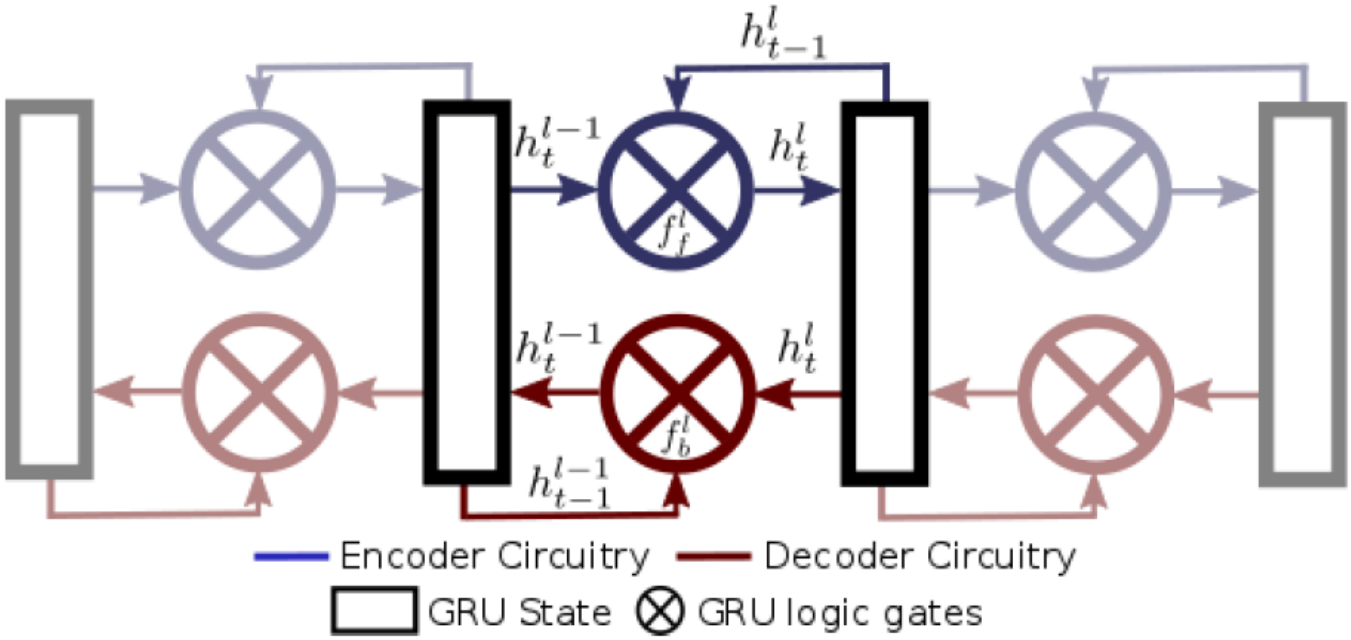
Proposal: Bijective GRU (bGRU)



Consider both input and output as recurrent states

Folded Recurrent Neural Networks

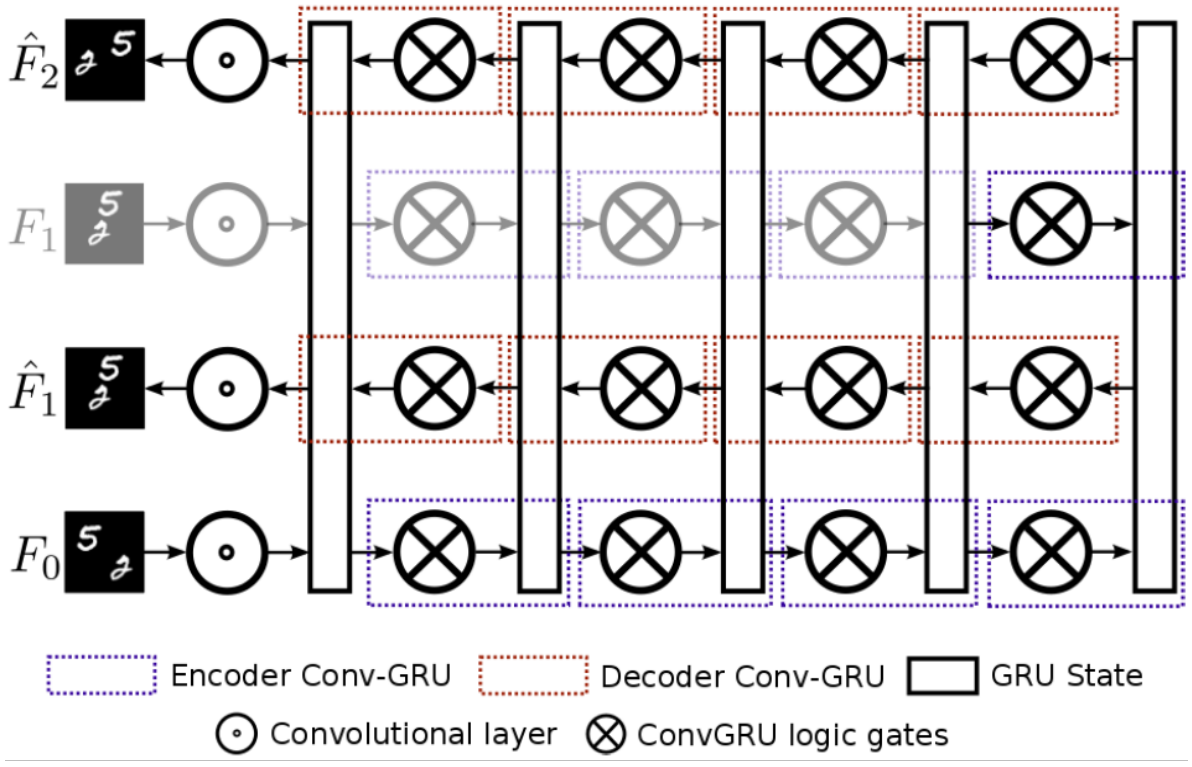
Proposal: Bijective GRU (bGRU)



Add an extra set of gates to update the input

Folded Recurrent Neural Networks

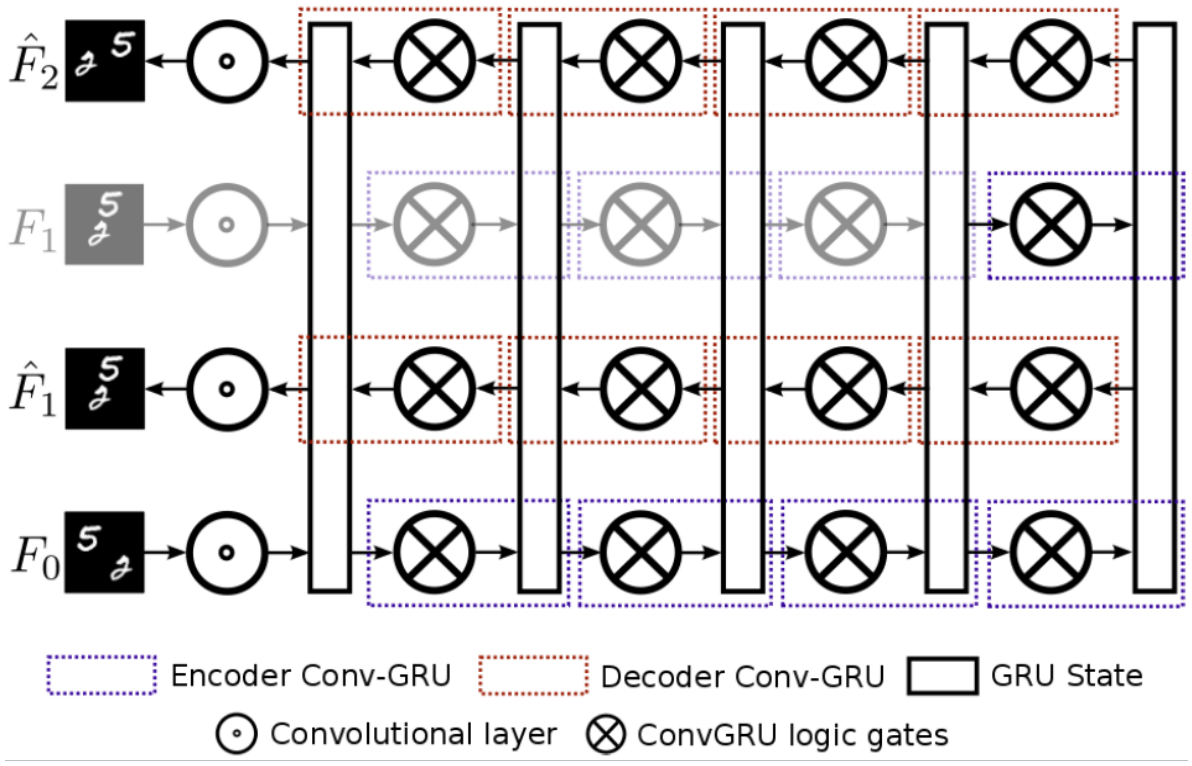
Proposal: Folded Recurrent Neural Networks (fRNN)



Lower cost: Use only the encoder/decoder at each step

Folded Recurrent Neural Networks

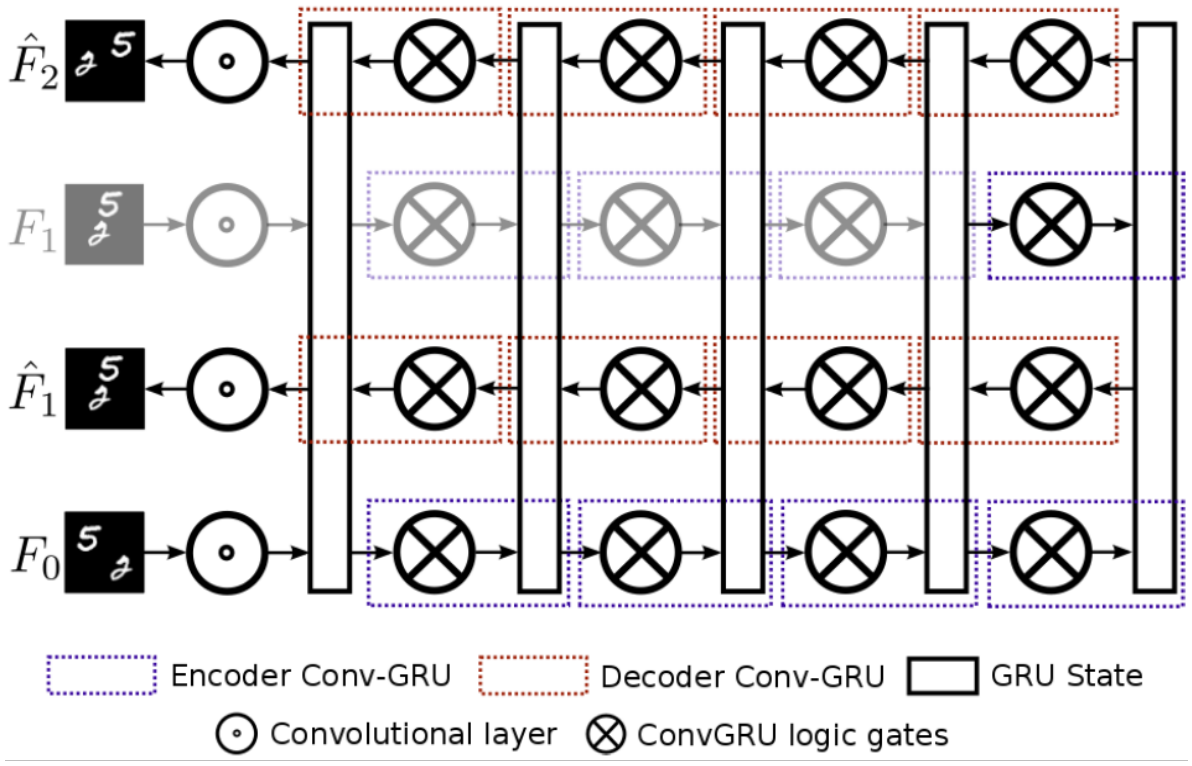
Proposal: Folded Recurrent Neural Networks (fRNN)



Prevents error propagation: No prediction re-encoding

Folded Recurrent Neural Networks

Proposal: Folded Recurrent Neural Networks (fRNN)



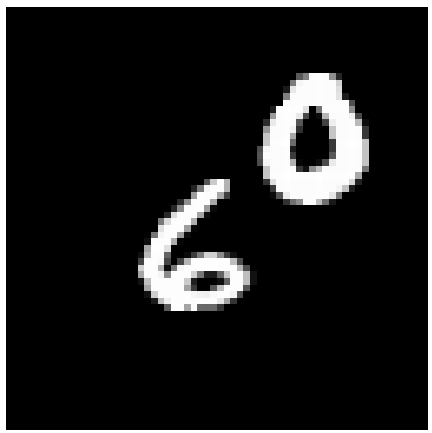
Less parameters: Implicit stratification

Experiments

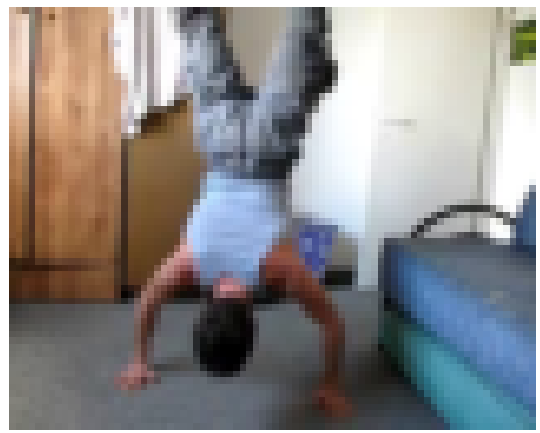
Parameters and datasets

	Conv 1	Conv 2	Pool 1	bGRU 1	bGRU 2	Pool 2	bGRU 3	bGRU 4	Pool 3	bGRU 5	bGRU 6	Pool 4	bGRU 7	bGRU 8
Num. Units	32	64	-	128	128	-	256	256	-	512	512	-	256	256
Kernel size	5×5	5×5	2×2	5×5	5×5	2×2	5×5	5×5	2×2	3×3	3×3	2×2	3×3	3×3
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	

KTH



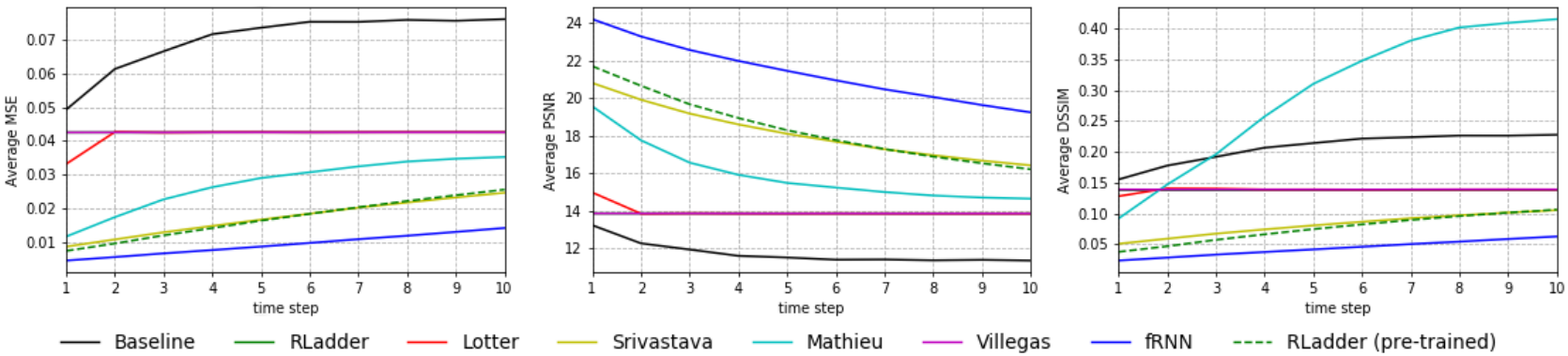
MMNIST



UCF-101

Experiments

Quantitative results (Moving MNIST)

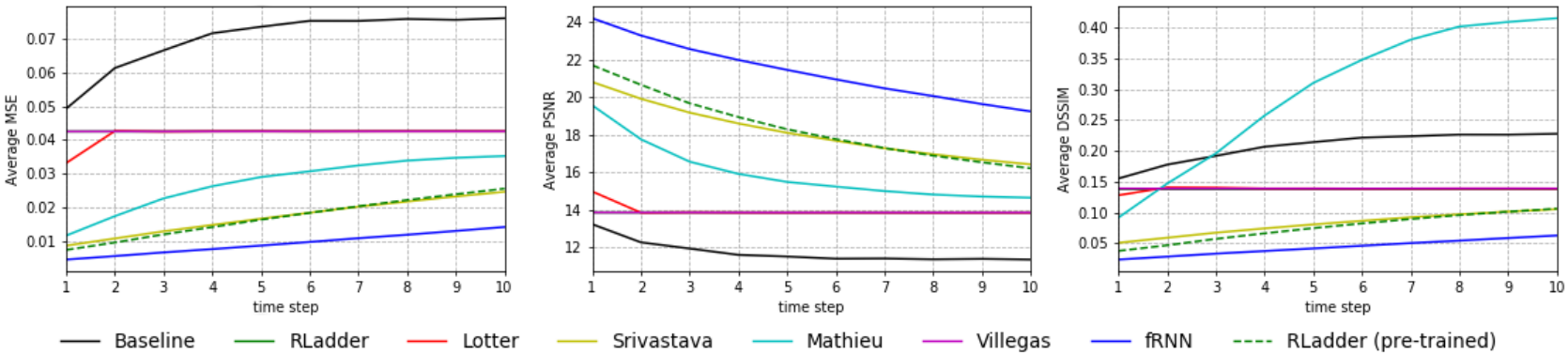


	MSE	PSNR	DSSIM
Baseline	0.06989	11.745	0.20718
RLadder	0.04254	13.857	0.13788
Lotter [2]	0.04161	13.968	0.13825
Srivastava [1]	0.01737	18.183	0.08164
Mathieu [23]	0.02748	15.969	0.29565
Villegas [11]	0.04254	13.857	0.13896
fRNN	**0.00947**	**21.386**	**0.04376**

On MMNIST it surpasses the state of the art and RLadder baseline by a good margin

Experiments

Quantitative results (Moving MNIST)

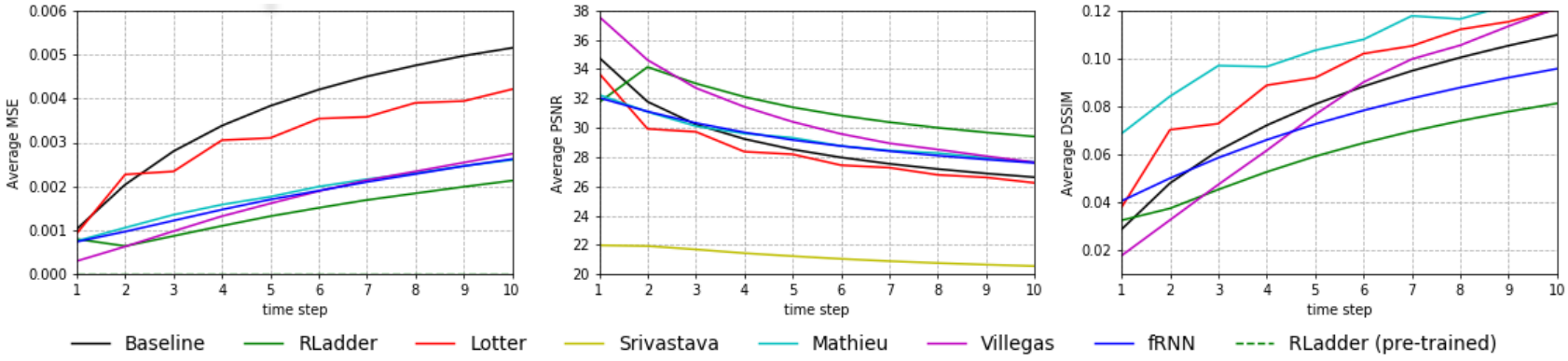


	MSE	PSNR	DSSIM
Baseline	0.06989	11.745	0.20718
RLadder	0.04254	13.857	0.13788
Lotter [2]	0.04161	13.968	0.13825
Srivastava [1]	0.01737	18.183	0.08164
Mathieu [23]	0.02748	15.969	0.29565
Villegas [11]	0.04254	13.857	0.13896
fRNN	**0.00947**	**21.386**	**0.04376**

Implicitly provides an identity mapping, preventing convergence problems

Experiments

Quantitative results (KTH)

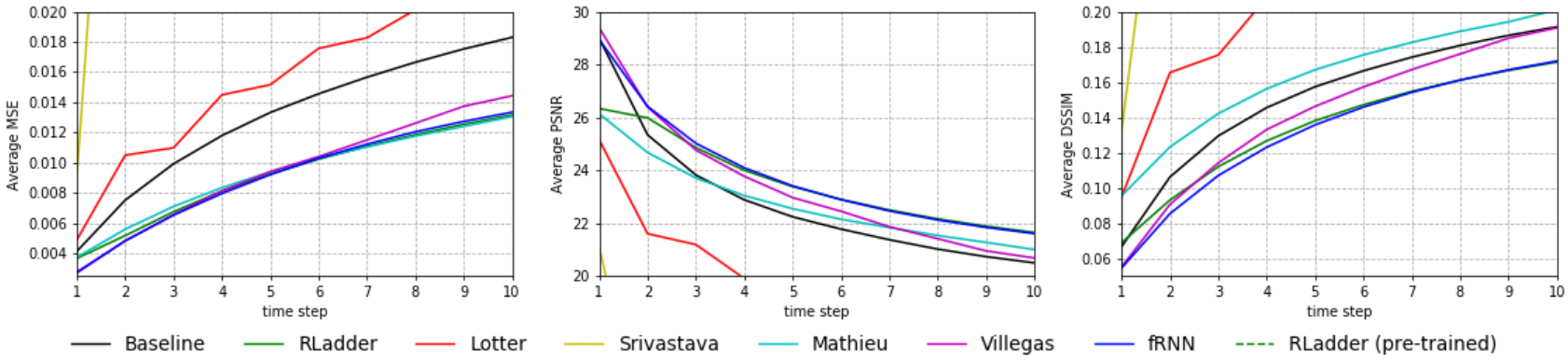


	MSE	PSNR	DSSIM
Baseline	0.00366	29.071	0.07900
RLadder	**0.00139**	**31.268**	**0.05945**
Lotter [2]	0.00309	28.424	0.09170
Srivastava [1]	0.00995	21.220	0.19860
Mathieu [23]	0.00180	29.341	0.10410
Villegas [11]	0.00165	30.946	0.07657
fRNN	0.00175	29.299	0.07251

On KTH it is comparable to the state of the art, only surpassed by the RLadder baseline

Experiments

Quantitative results (UCF101)

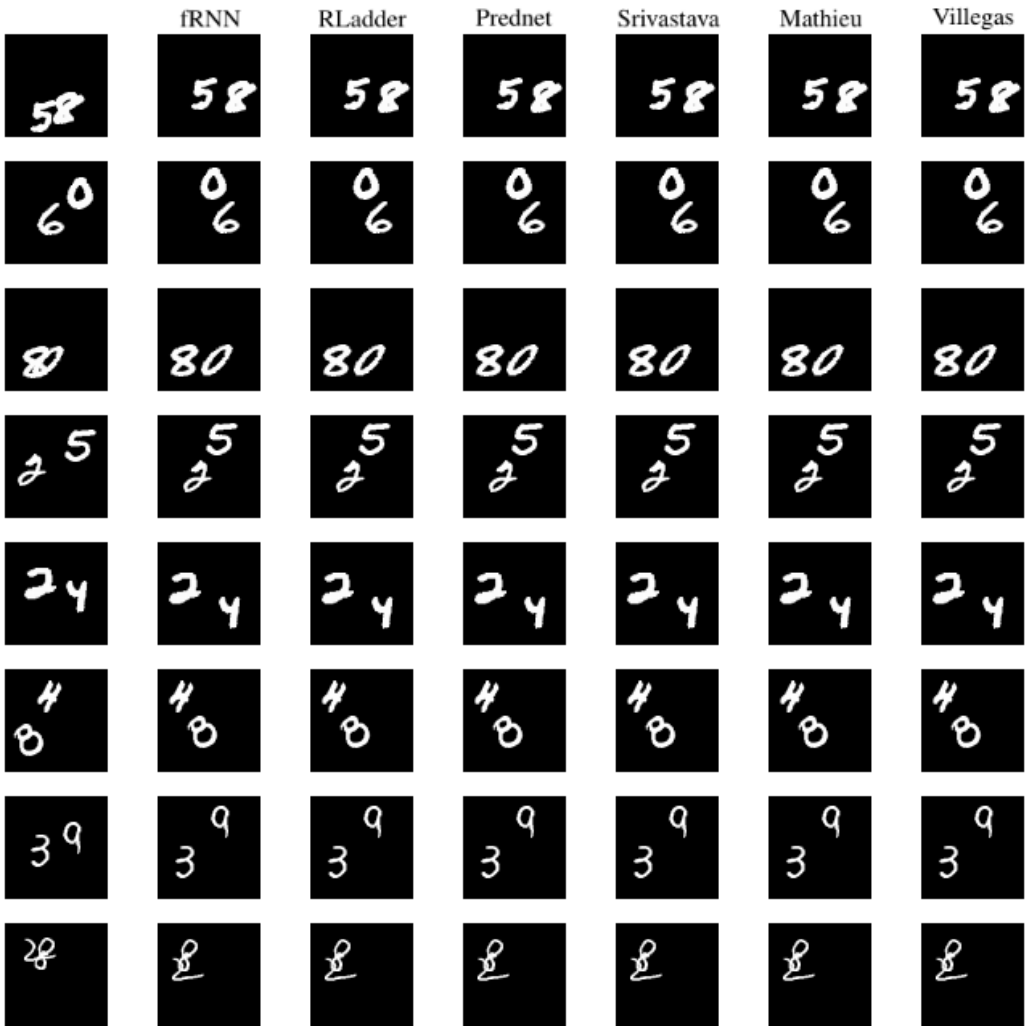


	MSE	PSNR	DSSIM
Baseline	0.01294	22.859	0.15043
RLadder	0.00918	23.558	0.13395
Lotter [2]	0.01550	19.869	0.21389
Srivastava [1]	0.14866	10.021	0.42555
Mathieu [23]	0.00926	22.781	0.16262
Villegas [11]	0.00940	23.457	0.14150
fRNN	**0.00908**	**23.872**	**0.13055**

On UCF101 (the most complex dataset) it obtains the best results

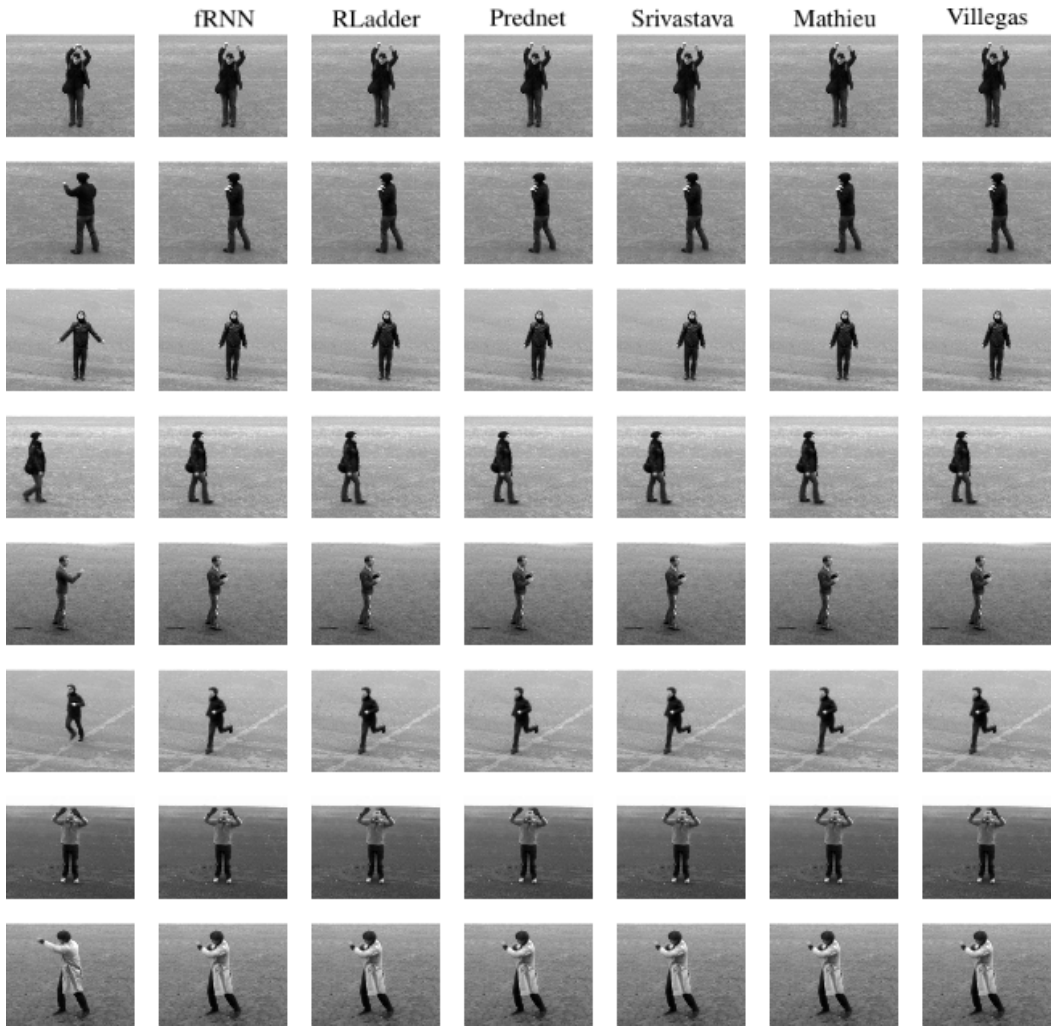
Experiments

Qualitative results (Moving MNIST)



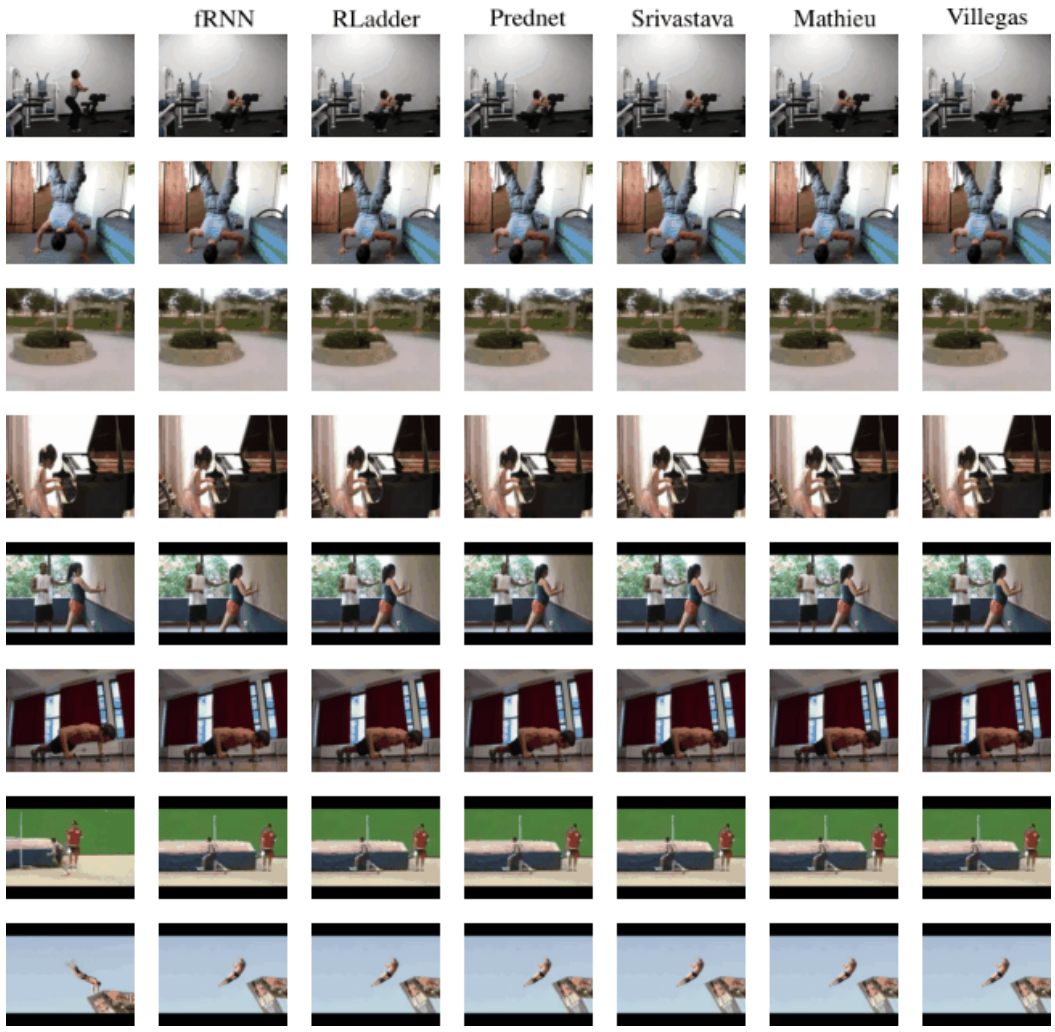
Experiments

Qualitative results (KTH)



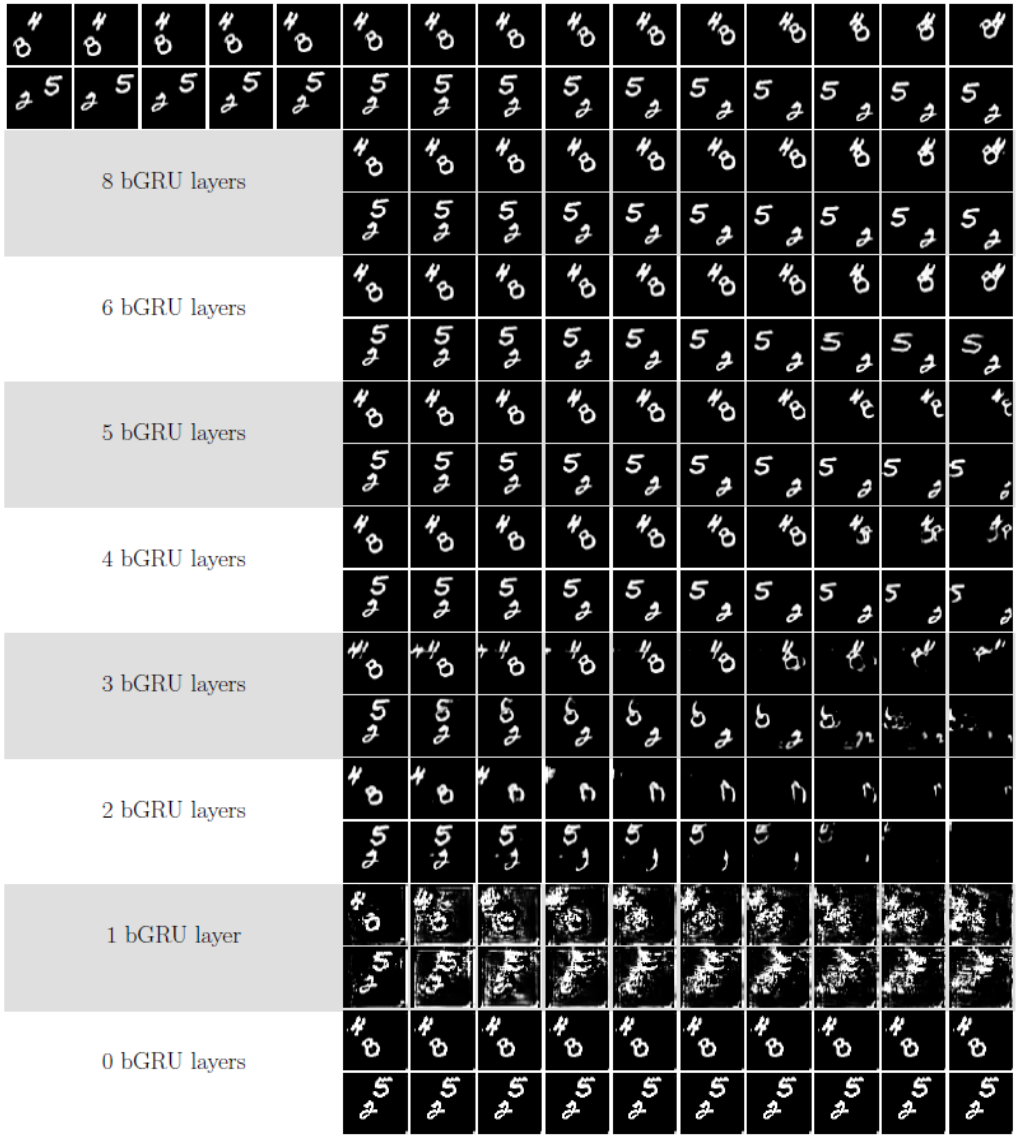
Experiments

Qualitative results (UCF101)



Experiments

Representation stratification



Conclusions

Contributions

- Greatly reduced error propagation through time
- Adaptive network depth
- Easy stratification analysis

Strengths

- Good implicit initialization
- Smaller memory footprint
- Smaller computational cost

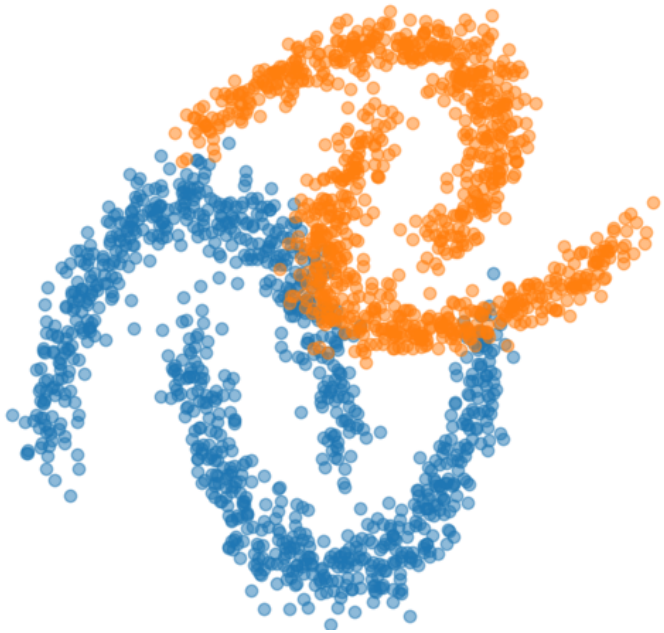
Cumulative Distribution Estimation

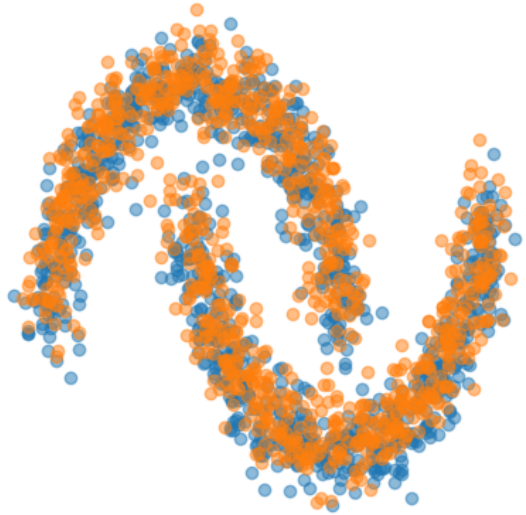
Multi-varied Cumulative Alignment

Domain Adaptation

The domain shift problem

Domain Adaptation consists on adapting an algorithm trained in one or more **source domains** to a related **target domain** sharing the same feature space but different sample distributions. The distribution discrepancy is known as **domain shift**.

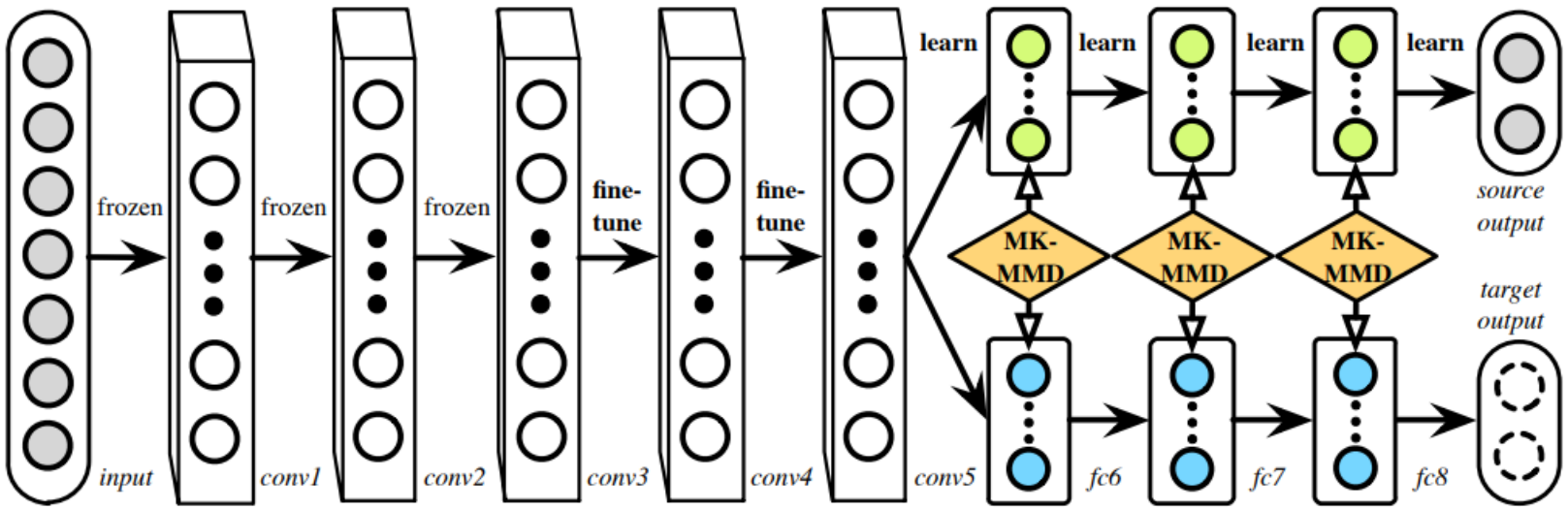




Domain Adaptation

Non-parametric distribution matching

In non-parametric models, a **metric** is used to minimize the discrepancy between both domains.

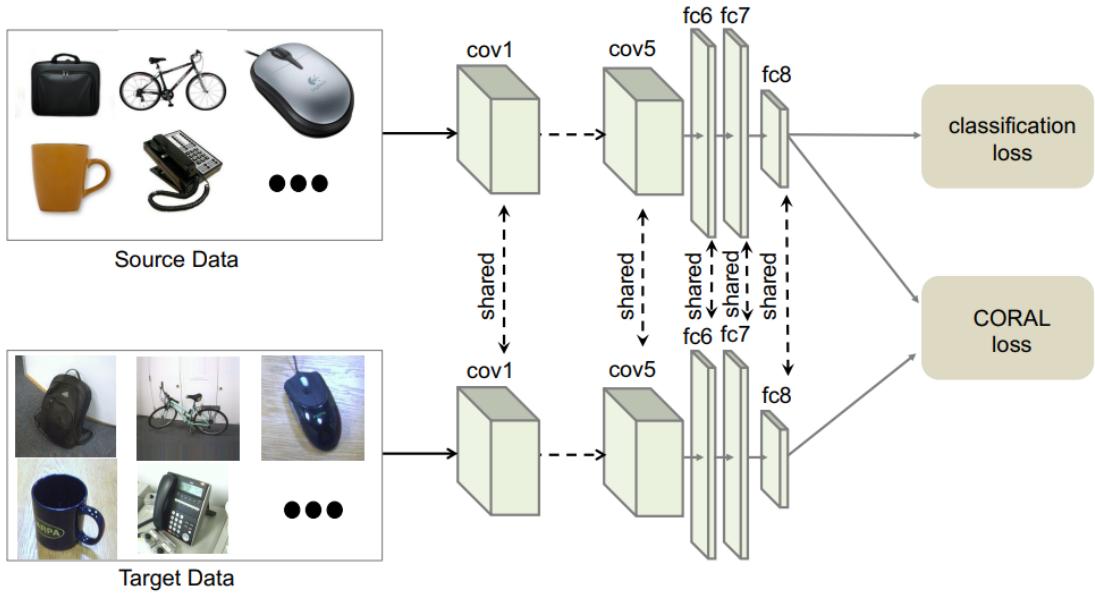


Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). *Learning transferable features with deep adaptation networks.* In ICML (97-105).

Domain Adaptation

Non-parametric distribution matching

In non-parametric models, a **metric** is used to minimize the discrepancy between both domains.

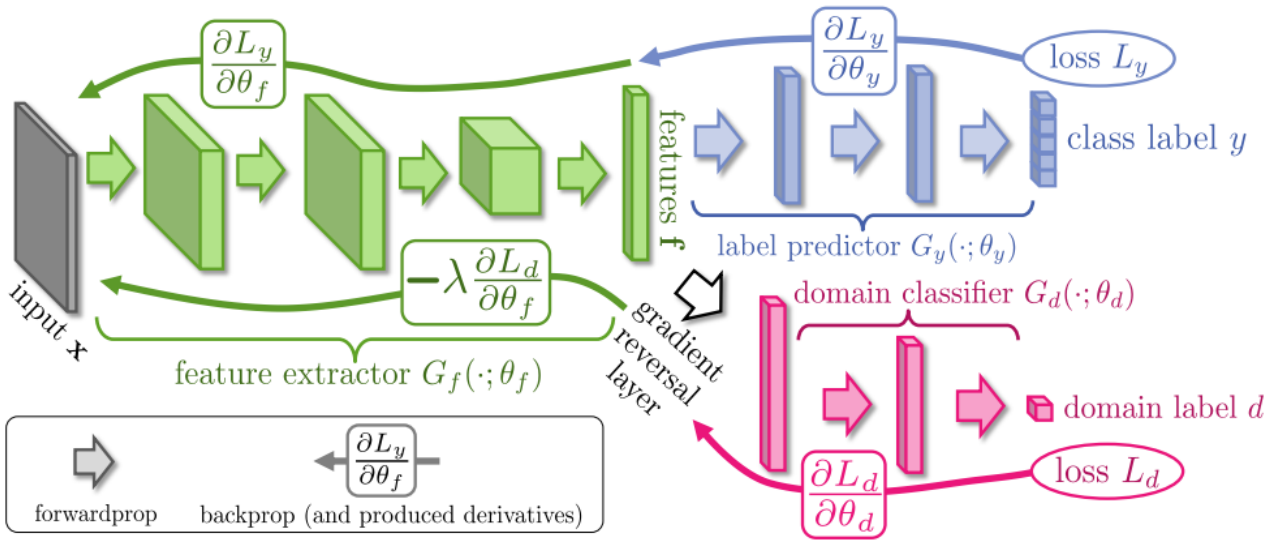


Sun, B., & Saenko, K. (2016). *Deep coral: Correlation alignment for deep domain adaptation.* In ECCV (443-450).

Domain Adaptation

Parametric distribution matching

In parametric models, a **non-linear model** is used to model both probability distributions, then minimize the discrepancies.

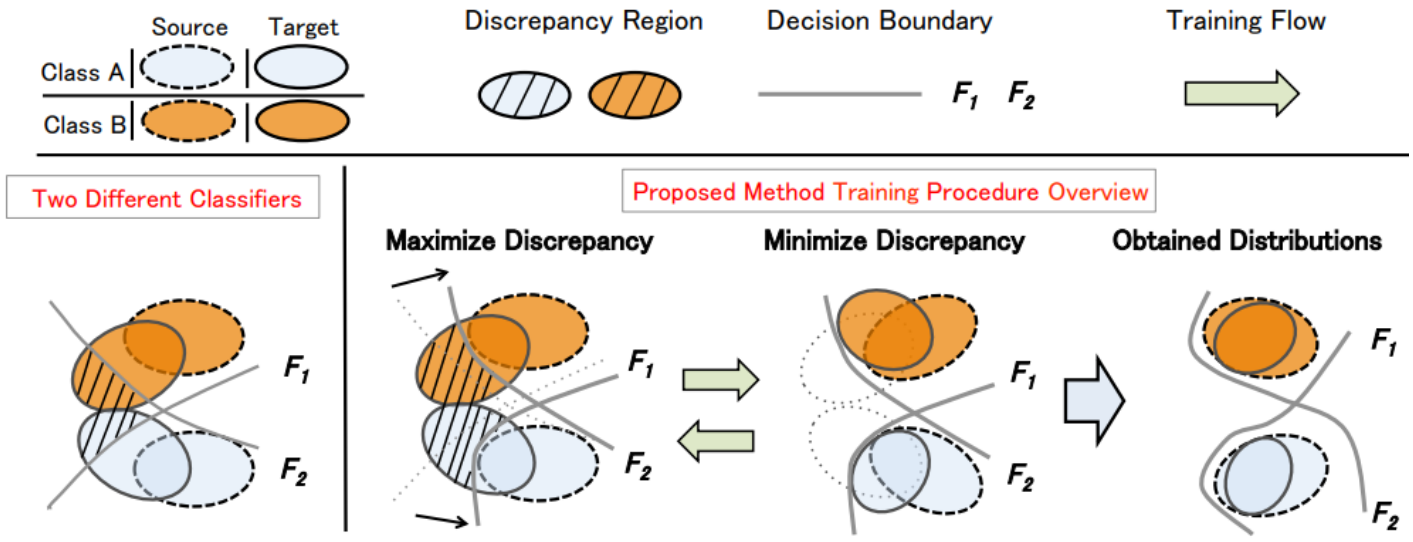


Ganin, Y., & Lempitsky, V. (2015). *Unsupervised domain adaptation by backpropagation.* In PMLR (1180-1189).

Domain Adaptation

Parametric distribution matching

In parametric models, a **non-linear model** is used to model both probability distributions, then minimize the discrepancies.



Saito, K., Watanabe, K., Ushiku, Y., & Harada, T. (2018). *Maximum classifier discrepancy for unsupervised domain adaptation.* In CVPR (3723-3732).

Domain Adaptation

Problems and limitations

- Poor approximation of the PDFs
- Dependency on additional learnable parameters
- Highly complex training pipelines

While **non-parametric models** are simple and resource efficient, the PDFs are **poorly approximated**, generally only capturing a quadratic approximation of the distribution.

Domain Adaptation

Problems and limitations

- Poor approximation of the PDFs
- Dependency on additional learnable parameters
- Highly complex training pipelines

While **parametric models** accurately capture the internal structure of the PDFs, they have a **high memory overhead** in the form of additional learnable parameters.

Domain Adaptation
Problems and limitations

- Poor approximation of the PDFs
- Dependency on additional learnable parameters
- Highly complex training pipelines

While **parametric models** accurately capture the internal structure of the PDFs, they have a **high computational overhead** in the form of both additional learnable parameters and more complex training pipelines.

Multi-varied Cumulative Alignment

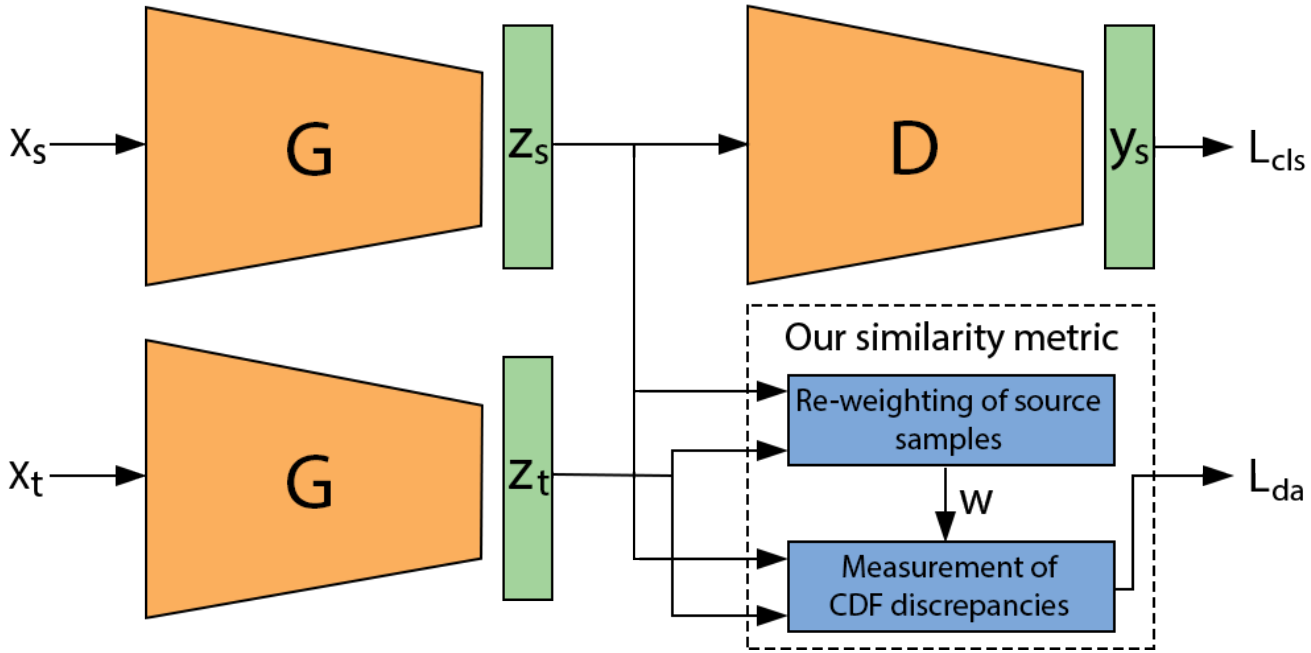
Proposal: Outline

We propose an approach capable of **aligning complex PDFs without directly modeling the distributions** themselves, resulting in a non-parametric model with the advantages of parametric approaches. We do so by stochastically sampling and aligning the projected CDFs of both domains.

- The internal structure of the PDFs is taken into account
- No additional trainable parameters are required
- Built-in solution to cross-domain distribution imbalances
- Not limited to classification tasks

Multi-varied Cumulative Alignment

Proposal: General pipeline



- Simple feature extractor + discriminator pipeline
- **Metric:** Re-weighting of source samples
- **Metric:** Measurement of CDF discrepancies

Multi-varied Cumulative Alignment

Proposal: Re-weighting of source samples

Both domains may (and usually will) suffer from **sample imbalances**. In order to correctly align both distributions, we'll first want to mitigate this issue. We'll do so by **re-weighting the source samples** in order to more closely match the target distribution.

$$\arg\min_{w} || \frac{1}{N_S} \widetilde{X}_S^T D_w \widetilde{X}_S - \frac{1}{N_T} \widetilde{X}_T^T \widetilde{X}_T ||_F^2$$

Goal: Find the source sample weights w minimizing the discrepancy between both domains' covariance matrices.

Multi-varied Cumulative Alignment

Proposal: Re-weighting of source samples

Both domains may (and usually will) suffer from **sample imbalances**. In order to correctly align both distributions, we'll first want to mitigate this issue. We'll do so by **re-weighting the source samples** in order to more closely match the target distribution.

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

Solution: Compact and efficient, with the matrix to invert being only of size $N_S \times N_S$.

Multi-varied Cumulative Alignment

Proposal: Re-weighting of source samples

Iteratively apply the weighting algorithm in order to better estimate both the **source sample mean** and **sample re-weighting**.

Require: X_S, X_T, N_S, N_T

$\widetilde{X}_T \leftarrow X_T - \frac{1}{N_T} \mathbb{1}^{<1 \times N_T>} X_T$

$w \leftarrow \mathbb{1}^{<N_S \times 1>}$

for *iter* **in** $[0 \ldots k]$ **do**

  $\widetilde{X}_S \leftarrow X_S - \frac{1}{N_T} w^T X_S$

  $A \leftarrow \frac{1}{N_S} (\widetilde{X}_S \widetilde{X}_S^T)^{\circ 2}$

  $B \leftarrow \frac{1}{N_T} (\widetilde{X}_S \widetilde{X}_T^T)^{\circ 2}$

  $w \leftarrow A^{-1} B \mathbb{1}^{<N_T \times 1>}$

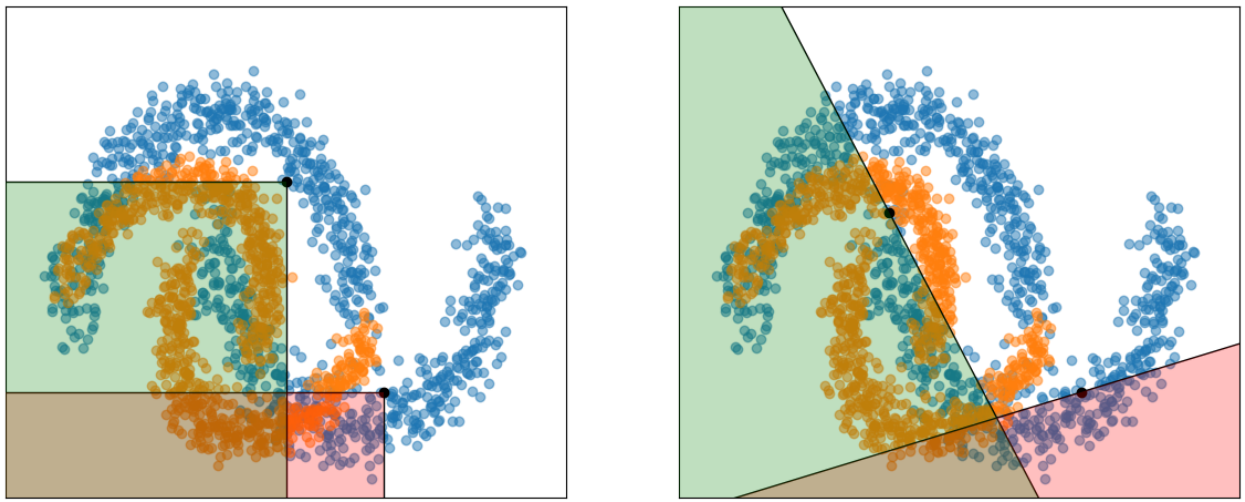
end for

$w \leftarrow \frac{N_S}{|w|_1} w$

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

Our **domain adaptation metric** is based on estimating the CDF at random points in both domains, then measuring the discrepancy across domains. This quickly runs into **under-sampling problems**.

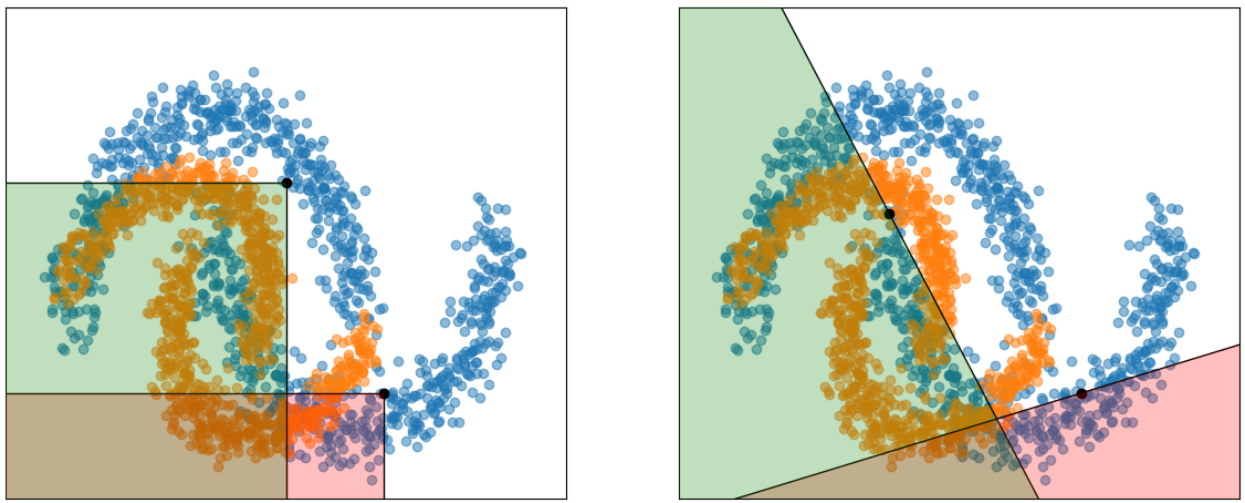


Left: The CDF estimate measures the ratio of samples in the negative octant of the feature space.

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

Our **domain adaptation metric** is based on estimating the CDF at random points in both domains, then measuring the discrepancy across domains. This quickly runs into **under-sampling problems**.



Right: The proposed projected CDF estimate measures the ratio of samples in the negative side of an hyper-plane.

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

We define $\widetilde{C}_s(x, v)$, $\widetilde{C}_t(x, v)$ as the functions estimating the value of the CDFs at point x and projected along vector v. Here, $\delta(x) \in \{0, 1\}$ is the sign function.

$$\widetilde{C}_S(x, v) = \frac{1}{N_S} \delta \left(X_S v - x^T v \right) \mathbb{1}^{<N_S \times 1>}$$

$$\widetilde{C}_T(x, v) = \frac{1}{N_T} \delta \left(X_T v - x^T v \right) \mathbb{1}^{<N_T \times 1>}$$

Note that $\delta(x)$ is **non-differentiable**, and thus not fit to use as a loss function.

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

We define $\widetilde{C}_s(x, v)$, $\widetilde{C}_t(x, v)$ as the functions estimating the value of the CDFs at point x and projected along vector v. Here, $\delta(x) \in \{0, 1\}$ is the sign function.

$$\widetilde{C}_S(x, v) = \frac{1}{N_S} \sigma \left(X_S v - x^T v \right) \mathbb{1}^{<N_S \times 1>}$$

$$\widetilde{C}_T(x, v) = \frac{1}{N_T} \sigma \left(X_T v - x^T v \right) \mathbb{1}^{<N_T \times 1>}$$

We first **replace** $\delta(x)$ **by a sigmoid function**, obtaining a differentiable approximation.

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

We define $\widetilde{C}_s(x, v)$, $\widetilde{C}_t(x, v)$ as the functions estimating the value of the CDFs at point x and projected along vector v. Here, $\delta(x) \in \{0, 1\}$ is the sign function.

$$\widetilde{C}_S(x, v) = \frac{1}{N_S} \sigma \left(X_S v - x^T v \right) w$$

$$\widetilde{C}_T(x, v) = \frac{1}{N_T} \sigma \left(X_T v - x^T v \right) \mathbb{1}^{<N_T \times 1>}$$

We then replace $\mathbb{1}^{<N_s \times 1>}$ by the weights vector w in order to **consider the source sample re-weighting**.

Multi-varied Cumulative Alignment

Proposal: Measurement of CDF discrepancies

The projected CDF estimates are aligned through **entropy maximization** over the **normalized estimates**.

$$L_{da} = \frac{1}{|P|} \sum_{(x,v)\in P} \overline{C}_S(x,v) log(\overline{C}_S(x,v)) + \overline{C}_T(x,v) log(\overline{C}_T(x,v))$$

$$\overline{C}_S(x,v) = \frac{\widetilde{C}_S(x,v)}{\widetilde{C}_S(x,v) + \widetilde{C}_T(x,v)}, \quad \overline{C}_T(x,v) = \frac{\widetilde{C}_T(x,v)}{\widetilde{C}_S(x,v) + \widetilde{C}_T(x,v)}$$

- $X_S \cup X_T$ are our sampling points
- Each sampling point is associated to multiple random v.

Multi-varied Cumulative Alignment

Proposal: Overall loss

The overall training loss is the weighted sum of the softmax classification loss L_{cls} and the domain adaptation loss L_{cls}.

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

The value for λ follows a sigmoid curve centered at zero, which results on a **monotonically increasing value** according to the following function:

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

In all our experiments, $\Lambda = 100$.

Experiments

Datasets: Digits

Three source-target domain pairs are considered for the digit classification task: SVHN to MNIST, MNIST to USPS, and USPS to MNIST.

MNIST consits of **binarized images** of hand-written digits. They display an homogeneous black background and no distractors.

Experiments

Datasets: Digits

Three source-target domain pairs are considered for the digit classification task: SVHN to MNIST, MNIST to USPS, and USPS to MNIST.

USPS introduces further variability in the form of **saturation changes** and **blur**.

Experiments

Datasets: Digits

Three source-target domain pairs are considered for the digit classification task: SVHN to MNIST, MNIST to USPS, and USPS to MNIST.

SVHN consists of real-world images taken from street digits. It introduces **color variability** and **distractors**.

Experiments

Datasets: Traffic signs

One target-domain pair is considered for the traffic sign recognition task: SYNSIG to GTSRB.

SYNSIG

GTSRB

The SYNthetic traffic SIGN recognition (SYNSIG) dataset consists of **synthetically generated** images of traffic signs spanning 40 different classes.

Experiments

Datasets: Traffic signs

One target-domain pair is considered for the traffic sign recognition task: SYNSIG to GTSRB.

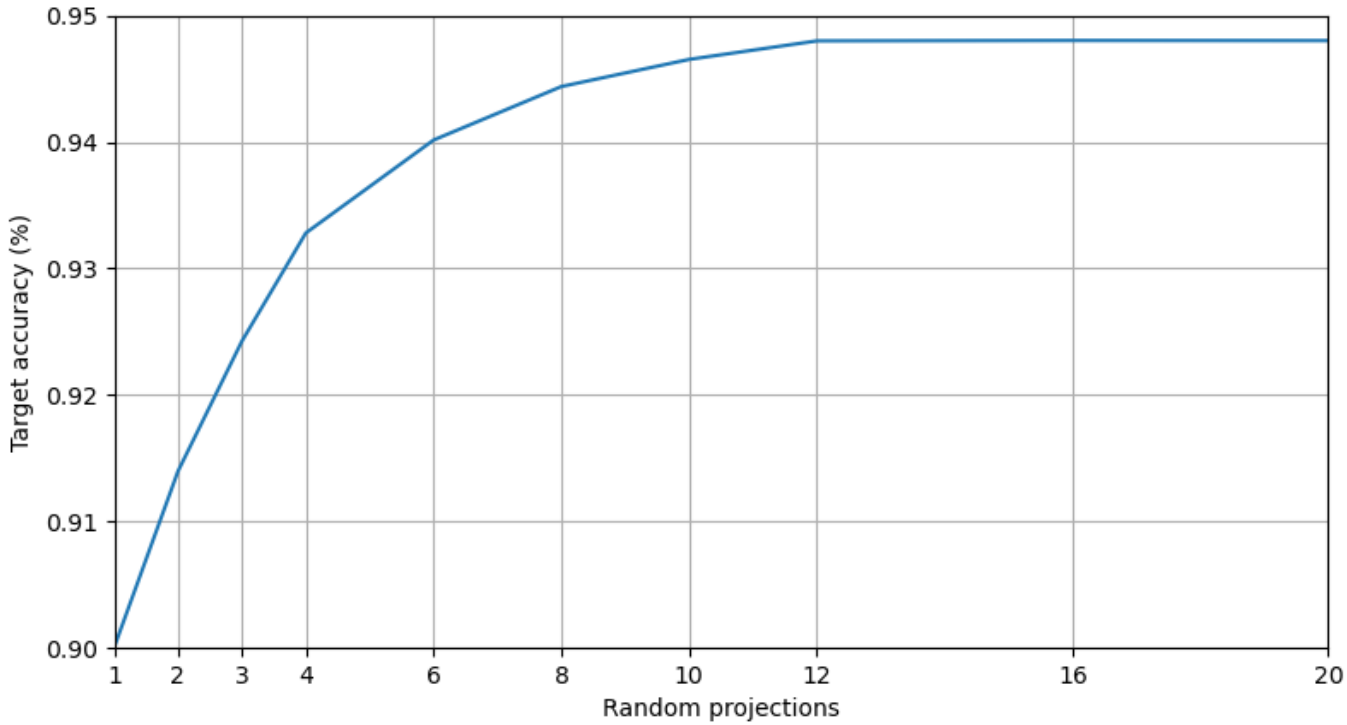
SYNSIG

GTSRB

The German Traffic Sign Recognition Benchmark (GTSRB) dataset consists of **real-world images** of traffic signs spanning 40 different classes.

Experiments

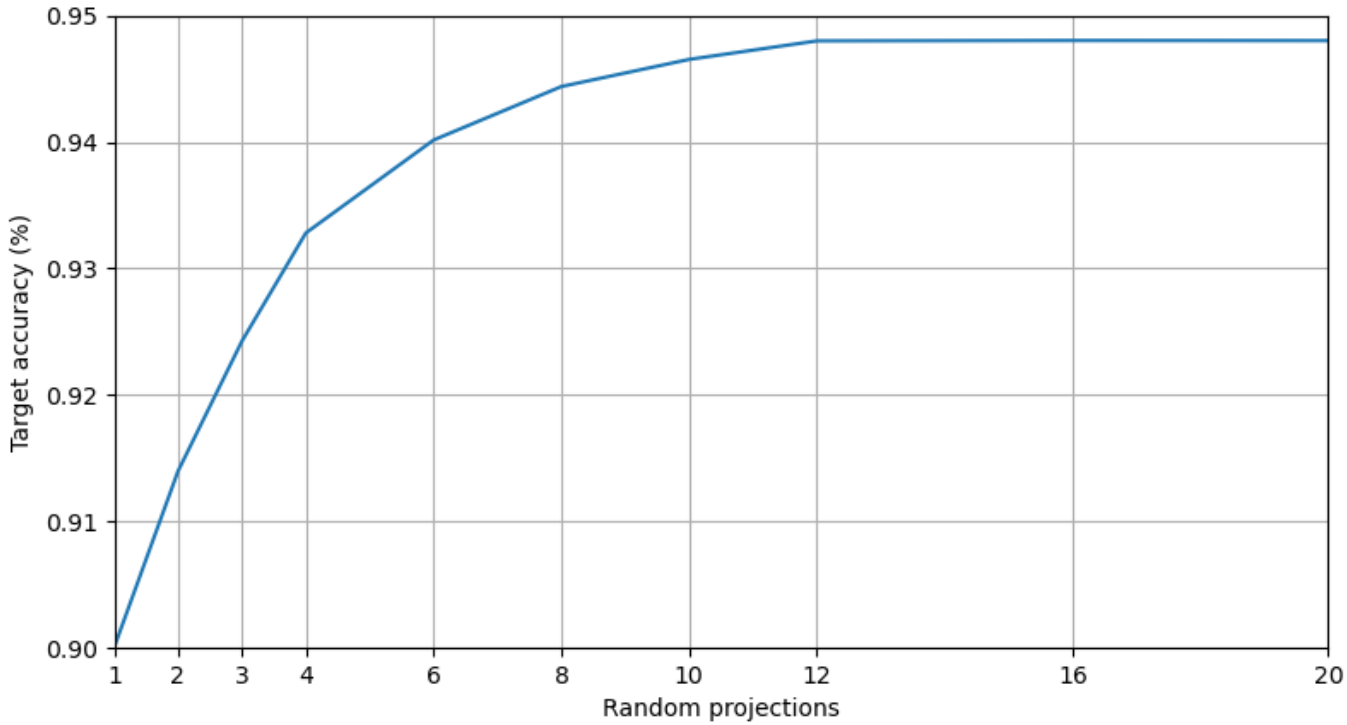
Calibrating the number of random projections



- Number of CDF projections per sample on SVHN to MNIST
- We performed 32 runs per value
- In all the experiments we used 12 random projections

Experiments

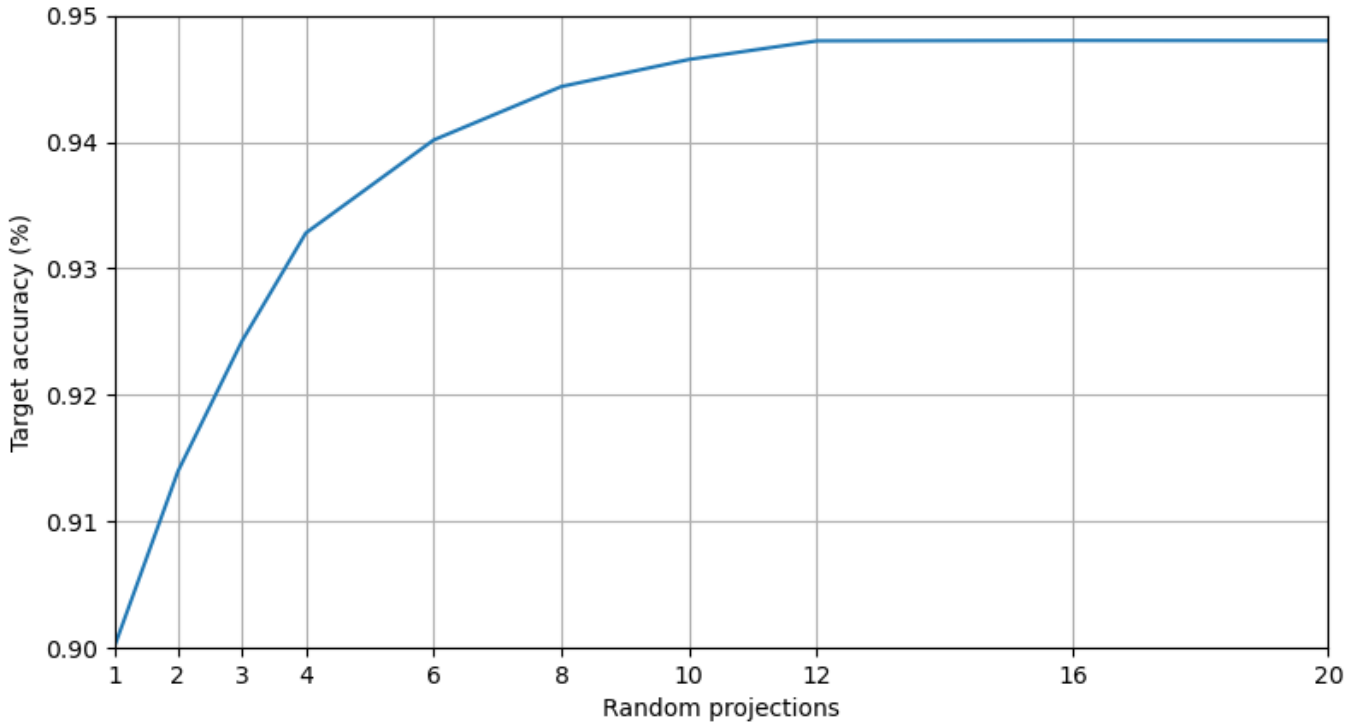
Calibrating the number of random projections



- Number of CDF projections per sample on SVHN to MNIST
- We performed 32 runs per value
- In all the experiments we used 12 random projections

Experiments

Calibrating the number of random projections



- Number of CDF projections per sample on SVHN to MNIST
- We performed 32 runs per value
- In all the experiments we used 12 random projections

Experiments

Comparison to the state-of-the-art

	SVHN to MNIST	**MNIST to USPS**	**USPS to MNIST**	**SYNSIG to GTSRB**
Source only	67.1%	79.4%	63.4%	85.1%
Non-parametric Distribution Matching methods				
MMD	71.1%	81.1%	-	91.1%
DeepCORAL	63.1%	80.7%	-	-
MCA (Ours)	**94.8%**	**94.6%**	**94.7%**	**95.1%**
Parametric Distribution Matching methods				
DAN	71.1%	85.1%	73.0%	88.7%
DSN	82.7%	91.3%	-	93.1%
MCD	96.2%	96.5%	94.1%	94.4%
SWD	**98.9%**	**98.1%**	**97.1%**	**98.6%**
DM-ADA	95.5%	96.7%	94.2%	-

The proposed method **outperforms any other non-parametric approach**, obtaining accuracies close to the state-of-the-art.

Experiments

Comparison to the state-of-the-art

	SVHN to MNIST	**MNIST to USPS**	**USPS to MNIST**	**SYNSIG to GTSRB**
Source only	67.1%	79.4%	63.4%	85.1%
Non-parametric Distribution Matching methods				
MMD	71.1%	81.1%	-	91.1%
DeepCORAL	63.1%	80.7%	-	-
MCA (Ours)	**94.8%**	**94.6%**	**94.7%**	**95.1%**
Parametric Distribution Matching methods				
DAN	71.1%	85.1%	73.0%	88.7%
DSN	82.7%	91.3%	-	93.1%
MCD	96.2%	96.5%	94.1%	94.4%
SWD	**98.9%**	**98.1%**	**97.1%**	**98.6%**
DM-ADA	95.5%	96.7%	94.2%	-

Only the parametric method by Lee *et al.* (SWD), published in 2019, consistently outperforms our proposal.

Conclusions

Contributions

- A new approach to solving cross-domain imbalances
- Method to align complex distributions without modeling them

Strengths

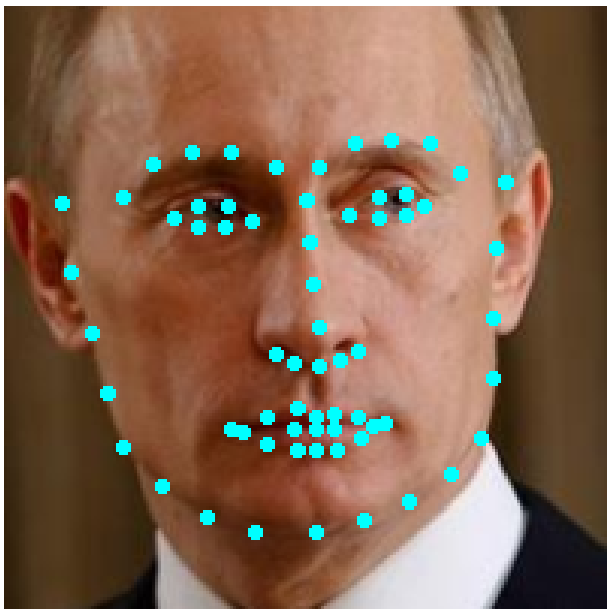
- No need for additional trainable parameters
- Forgoes adversarial training
- Amenable to regression tasks (in theory)

To conclude

Some final considerations

Summary

Contributions of the three approaches

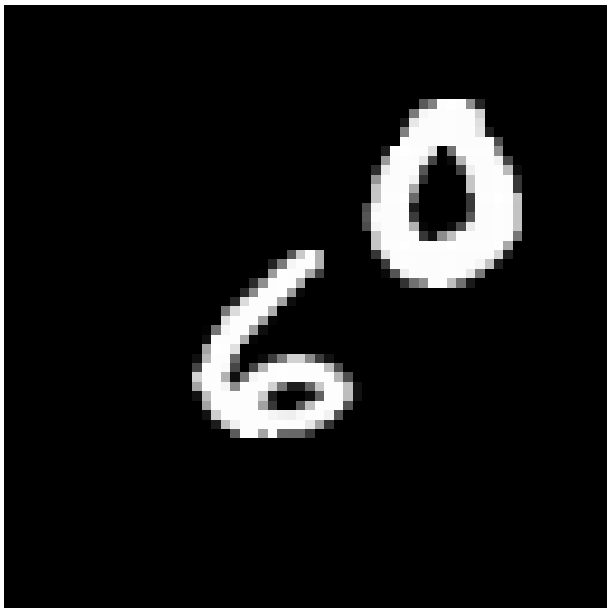


Just **increasing the order of a model** can achieve great results, so long as we are able to **limit the higher order polynomials** to the most relevant terms.

- Problem already solved

Summary

Contributions of the three approaches

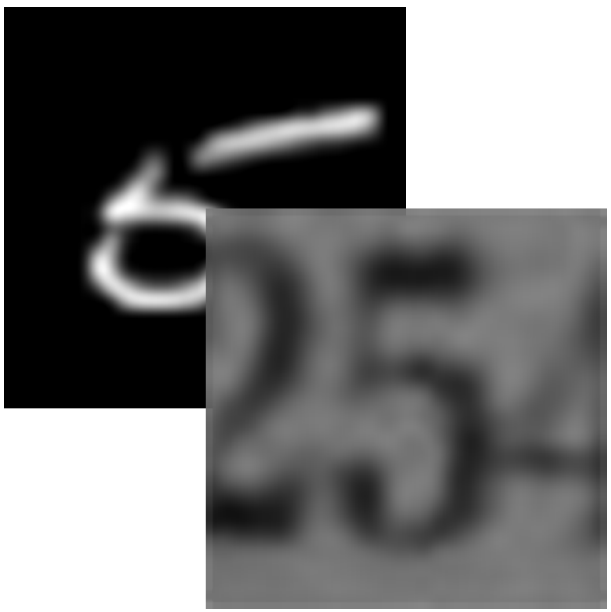


Taking into account the flow of information in order to **avoid clogging the high-order representations** with low-order information leads to benefits in both memory and computational resources.

- Difficulty with multiple futures
- More efficient models needed

Summary

Contributions of the three approaches



For certain tasks, we can **bypass modeling the high-order function altogether**. A series of stochastic evaluations of the same can take its place instead.

- Low accuracy for complex datasets
- Open problem for video sequences

Spatiotemporal analysis of RGB-DT facial images for multimodal pain level recognition
2015 CVPR Workshops

Chalearn looking at people 2015 new competitions: Age estimation and cultural event recognition
2015 IJCNN

Chalearn lap 2016: First round challenge on first impressions-dataset and results
2016 ECCV Workshops

Continuous Supervised Descent Method for Facial Landmark Localization
2016 ACCV

Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-related Applications
2016 TPAMI

Overcoming Calibration Problems in Pattern Labeling with Pairwise Ratings: Application to Personality Traits
2016 ECCV Workshops

Thank you

Improved RGB-DT based face recognition
2016 IET Biometrics

Chalearn looking at people and faces of the world: Face analysis workshop and challenge 2016
2016 CVPR Workshops

Method and station for the automatic quality control of multimodel biometric data
2017 ES 17382223

Folded Recurrent Neural Networks for Future Video Prediction
2018 ECCV

Multi-task human analysis in still images: 2D/3D pose, depth map, and multi-part segmentation
2019 FG

Multi-varied Cumulative Alignment for Domain Adaptation
2022 ICIAP