VALSE seminar September 2016

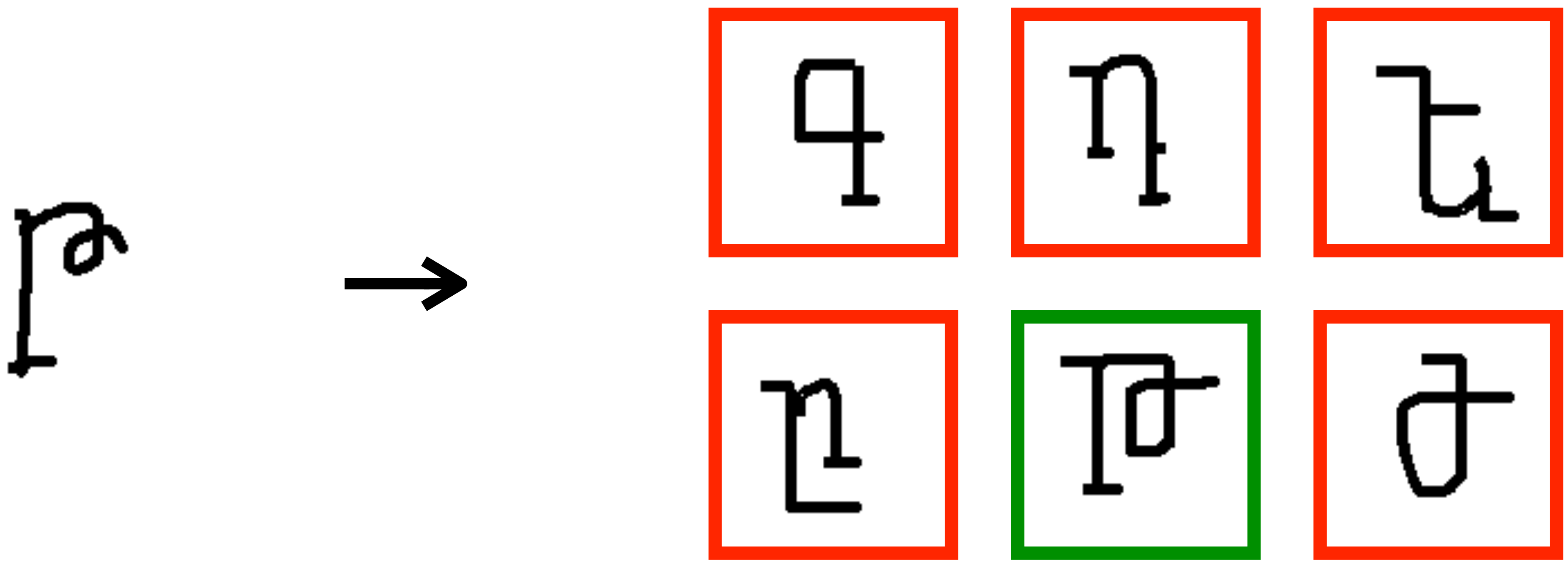
Learning feed-forward one-shot learners

NIPS 2016

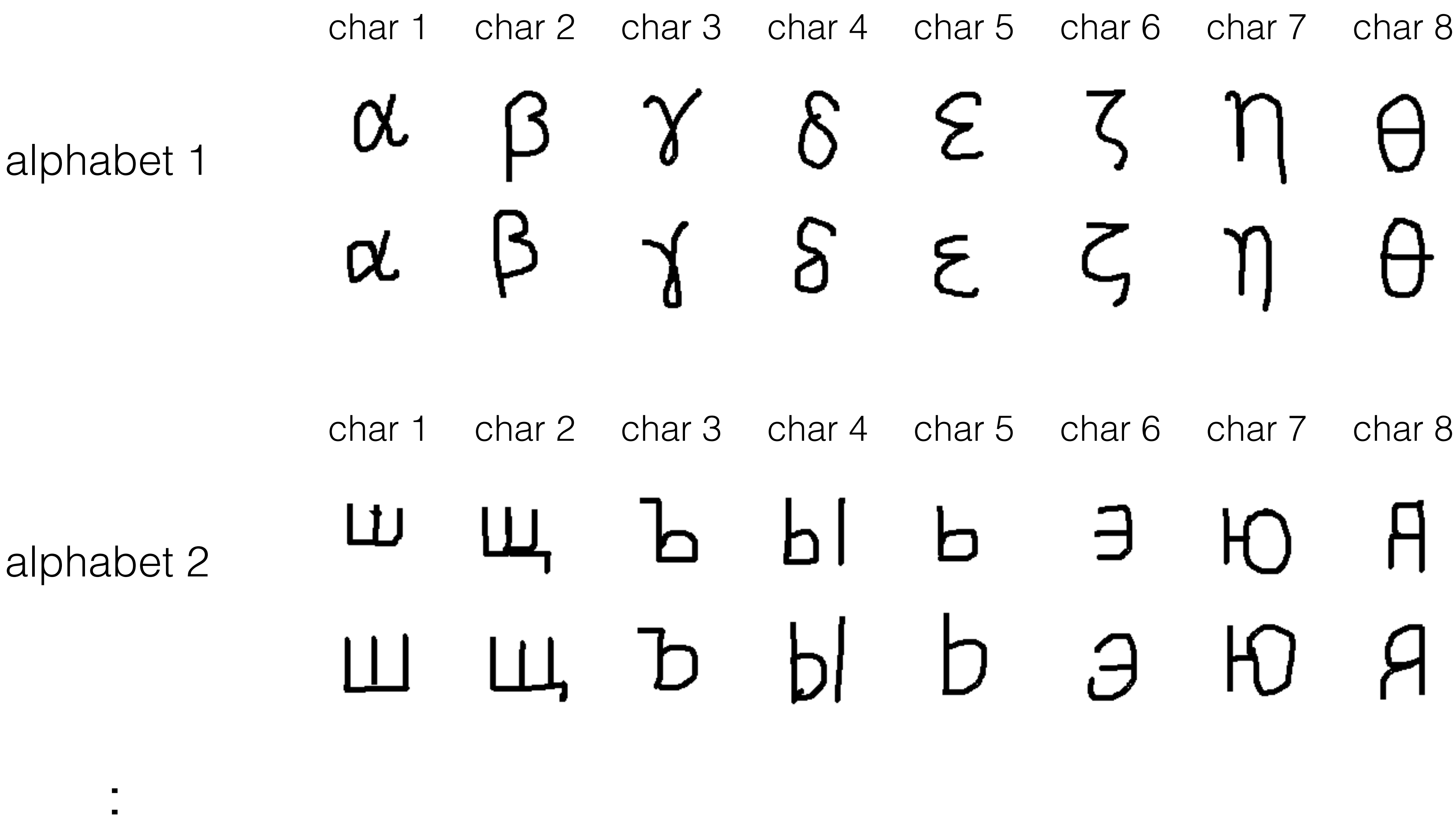
Bertinetto* Henriques* Valmadre* Torr Vedaldi
University of Oxford

One-shot learning

Learn a concept (classifier) from one example

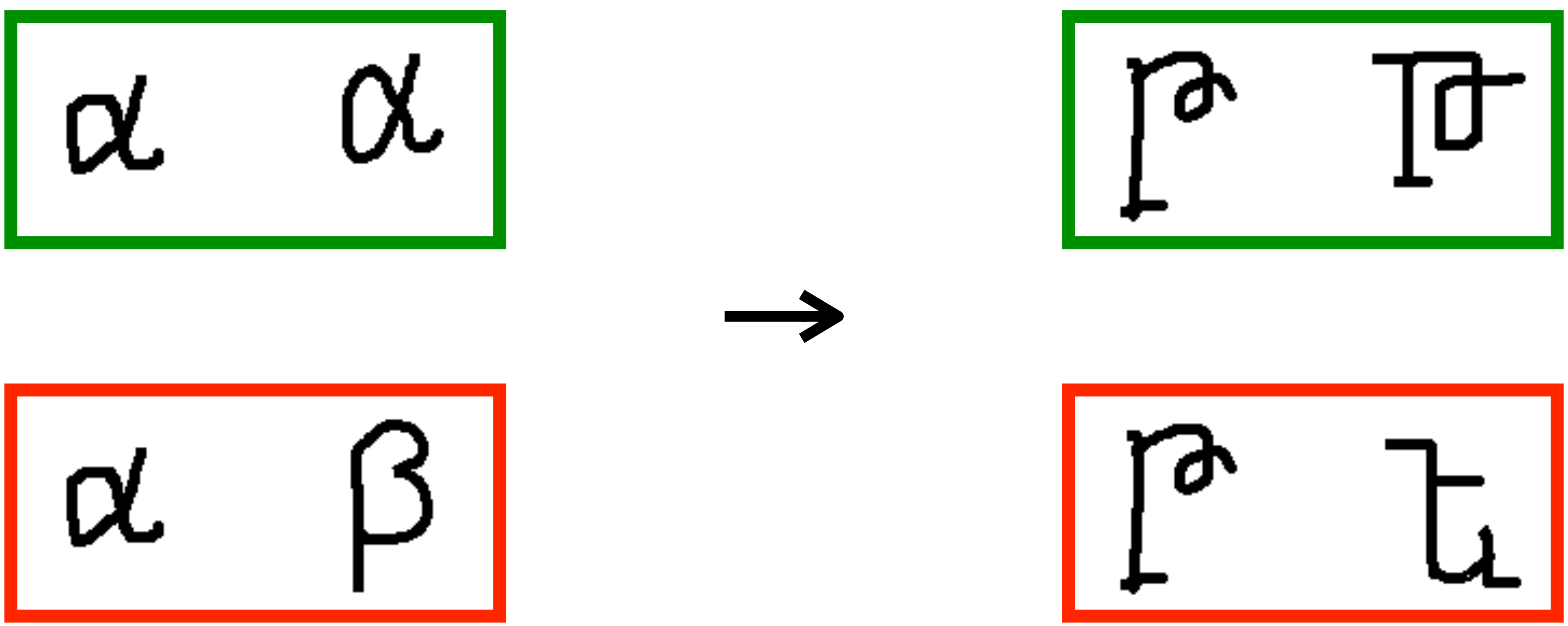


One-shot learning



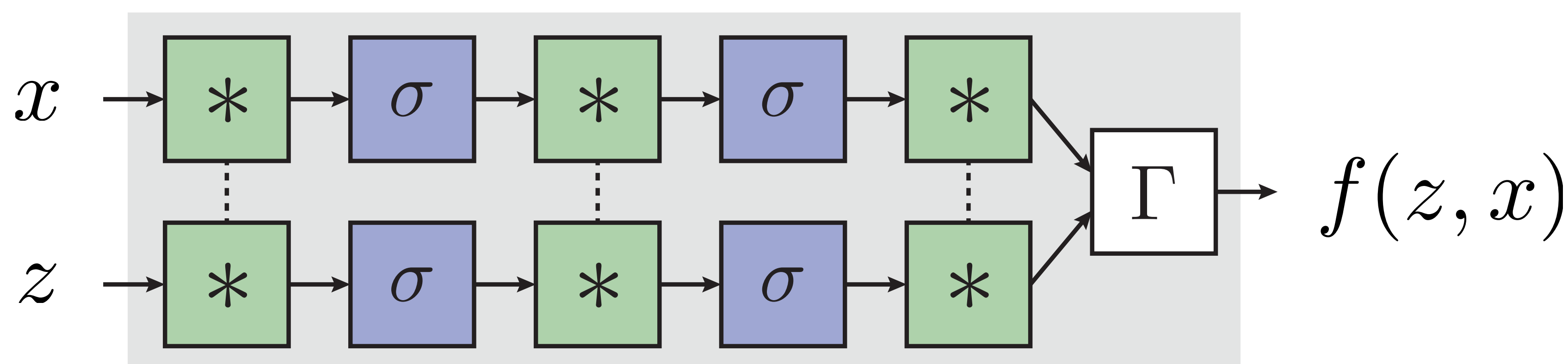
One-shot learning

Standard approach: similarity learning



One-shot learning

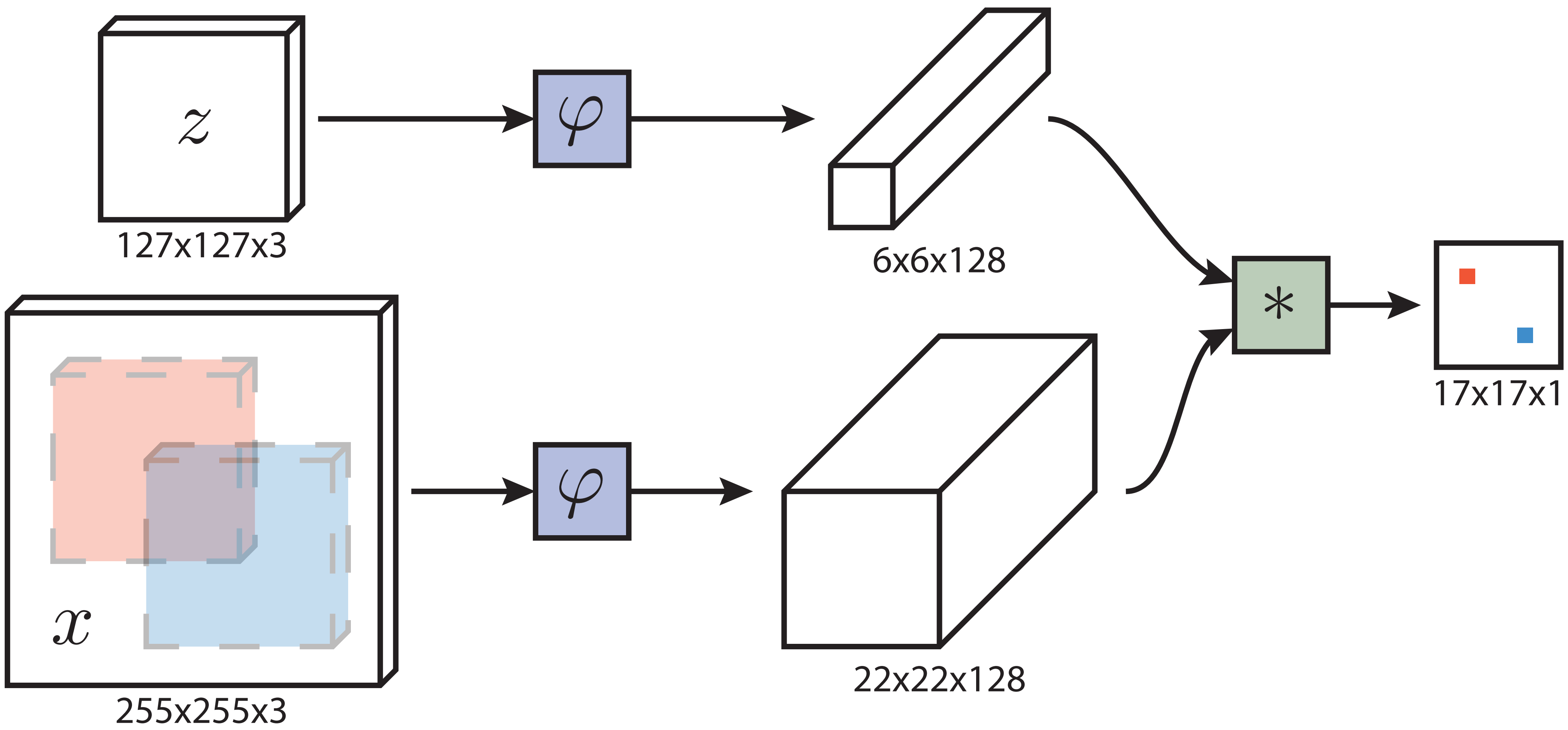
Natural choice: siamese network + logistic regression (same/different classifier)



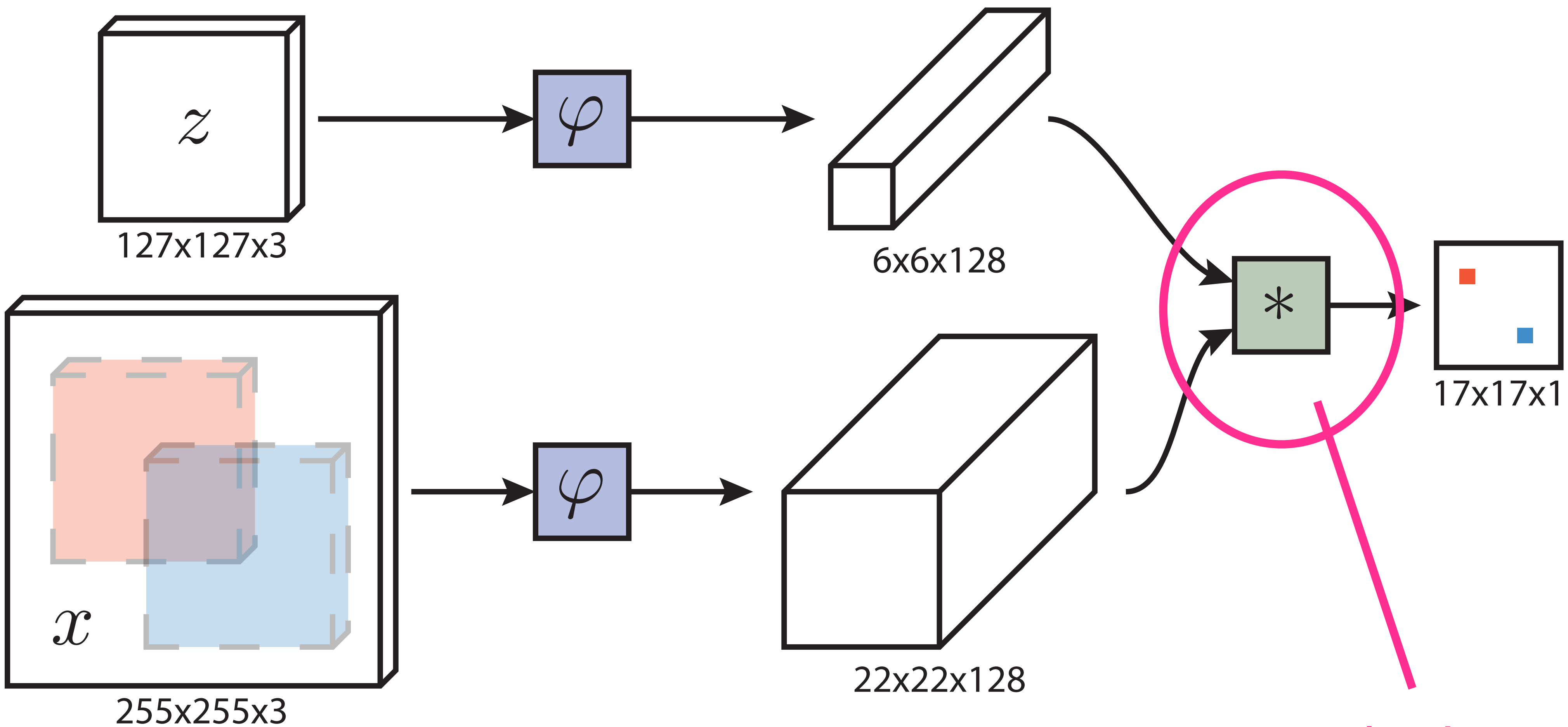
$$\min_{f} \mathop{\mathbb{E}}_{(z,x,y)} L(y, f(z,x))$$

z	x	y
α	α	+1
α	β	−1

Tracking as one-shot learning



Tracking as one-shot learning

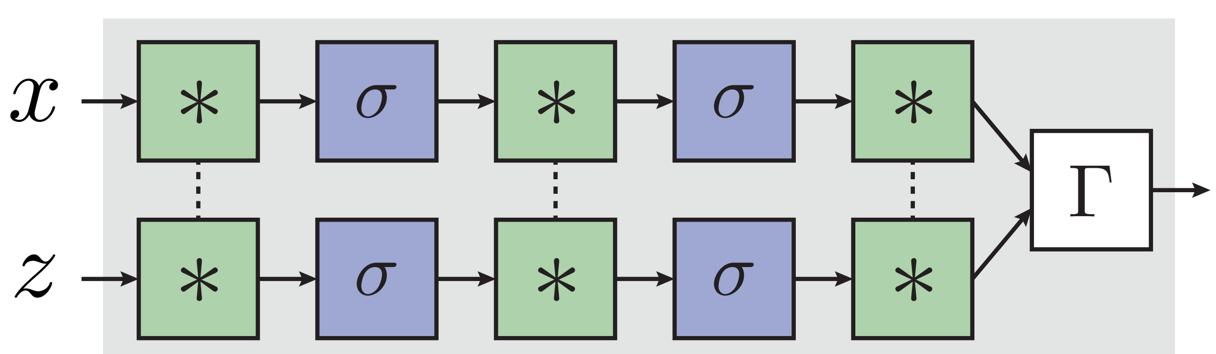


convolution with
a dynamic filter…

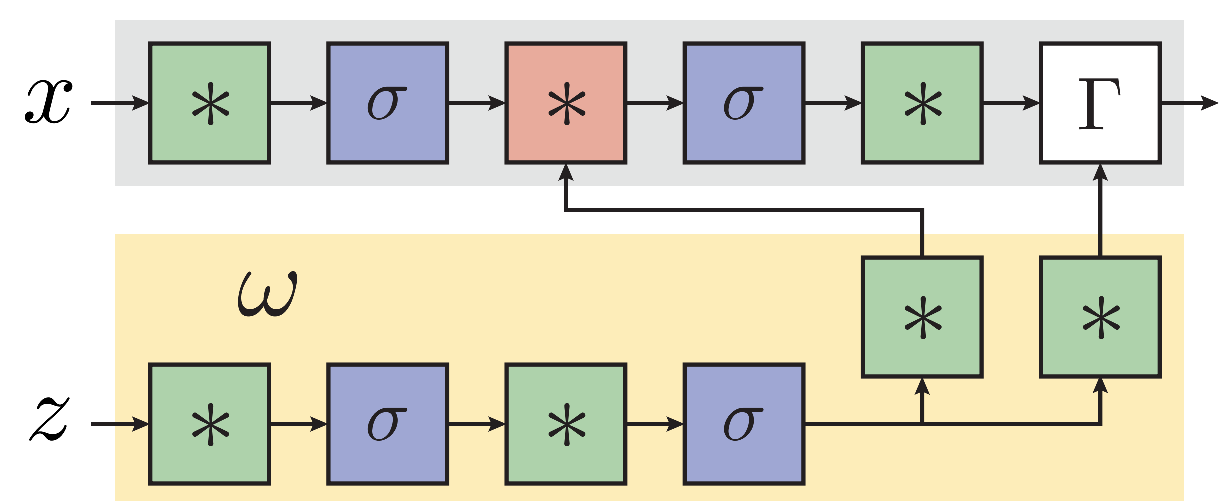
Learnets

siamese: $f(z,x) = \Gamma(\varphi(x; W),\ \varphi(z; W))$

learnet: $f(z,x) = \Gamma(\varphi(x; \omega(z; W')),\ \varphi(z; W))$



siamese



learnet

Learnets

Are learnets really “learning to learn”?

Learnets

Are learnets really “learning to learn”?

If we define learning as a procedure that maps a set of examples to a function, then yes

Practical difficulty: Output dimension

Typical number of parameters:

$4096 \times 4096 \approx 2e7$ for fully-connected

$3 \times 3 \times 192 \times 256 \approx 4e6$ for convolutional

Practical difficulty: Output dimension

Typical number of parameters:

4096 x 4096 ≈ 2e7 for fully-connected

3 x 3 x 192 x 256 ≈ 4e6 for convolutional

To predict this many outputs from a 4096-dim vector:

4096 x 4e6 ≈ 1e9 params (6.8GB of float32)

Factorisation: Fully-connected case

Inspired by SVD

$$Wx = U \, \mathrm{diag}(s) V^T x$$

Learn constant (non-orthogonal) basis and predict weights of diagonal transform

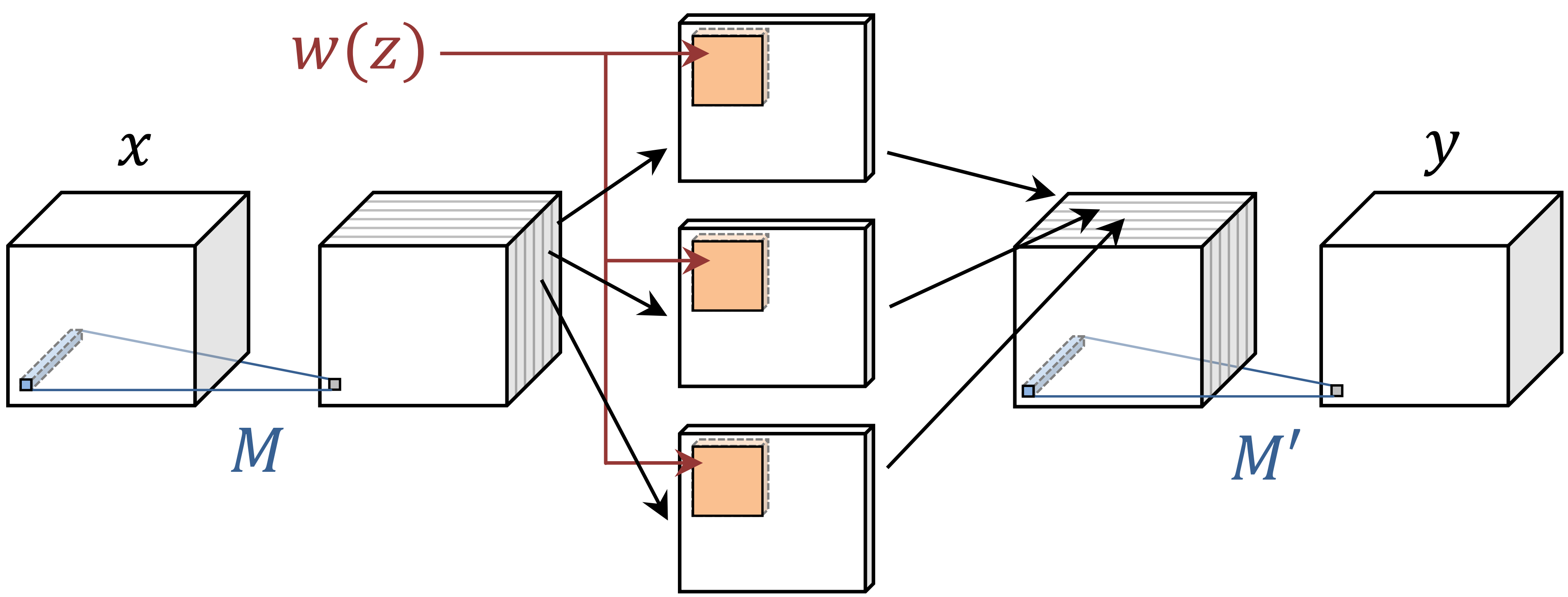
$$W(z) \cdot x = M' \, \mathrm{diag}(w(z)) M x$$

(Also predict bias)

Factorisation: Convolutional case

1×1 conv; m×m <u>diag conv</u> (dynamic); 1x1 conv

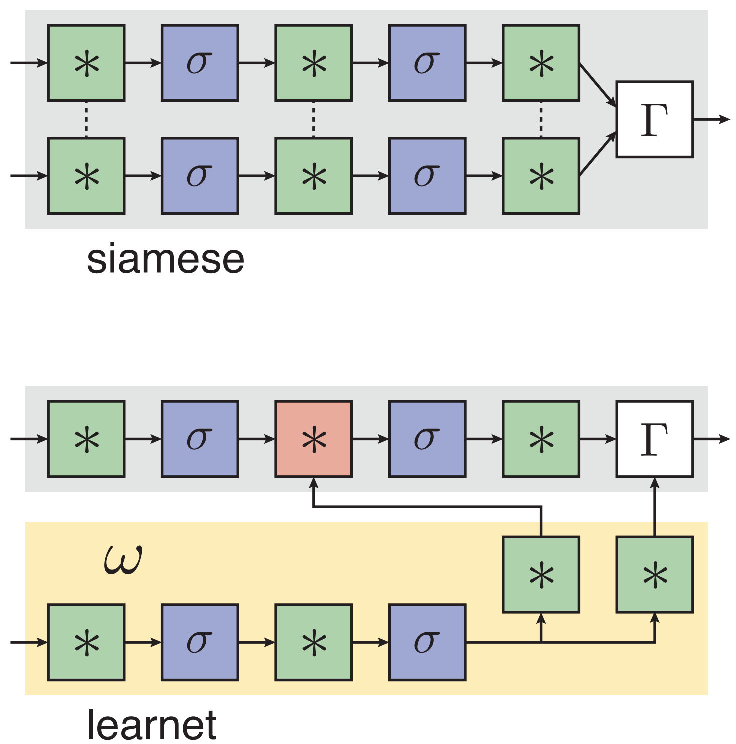
diag conv applies one filter to each channel

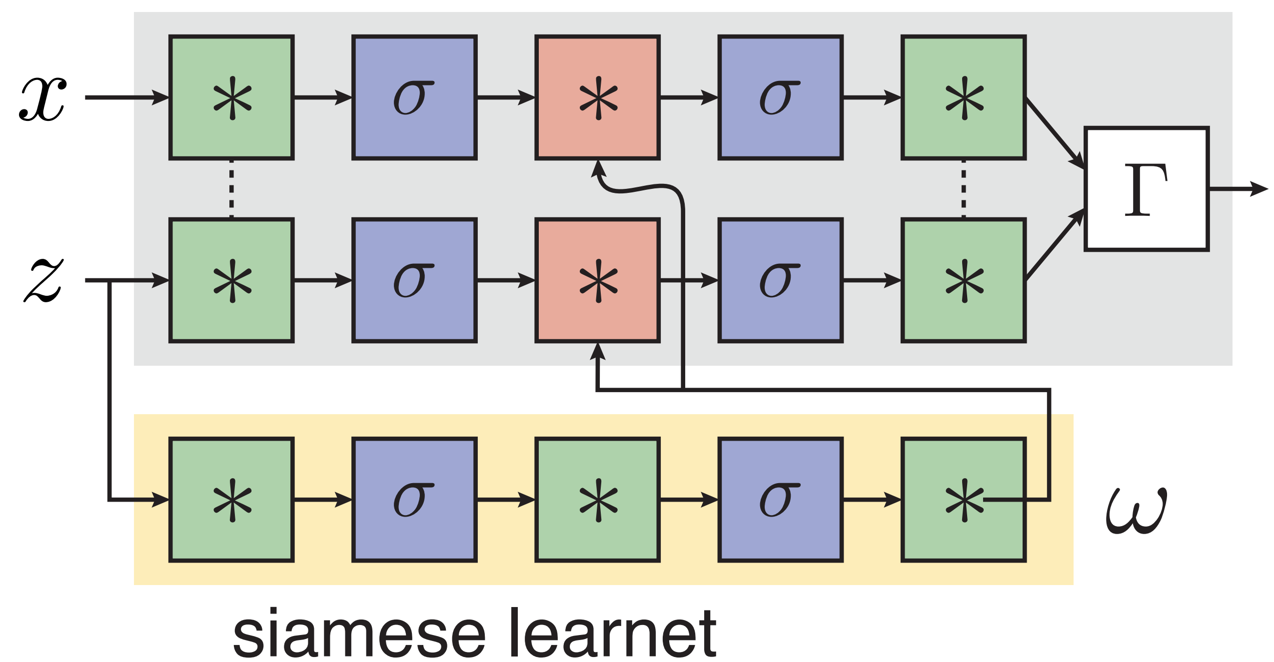


Implementation detail

- cudnn does not have an efficient primitive for diagonal convolution
- Implemented as dense convolution with zeros inserted in off-diagonal

Conditional embedding





$$f(z,x) = \Gamma(\varphi(x; \omega(z; W')),\ \varphi(z; \omega(z; W')))$$

Experiments

- Omniglot (proof of concept; 28×28 px; small net)
- Object tracking

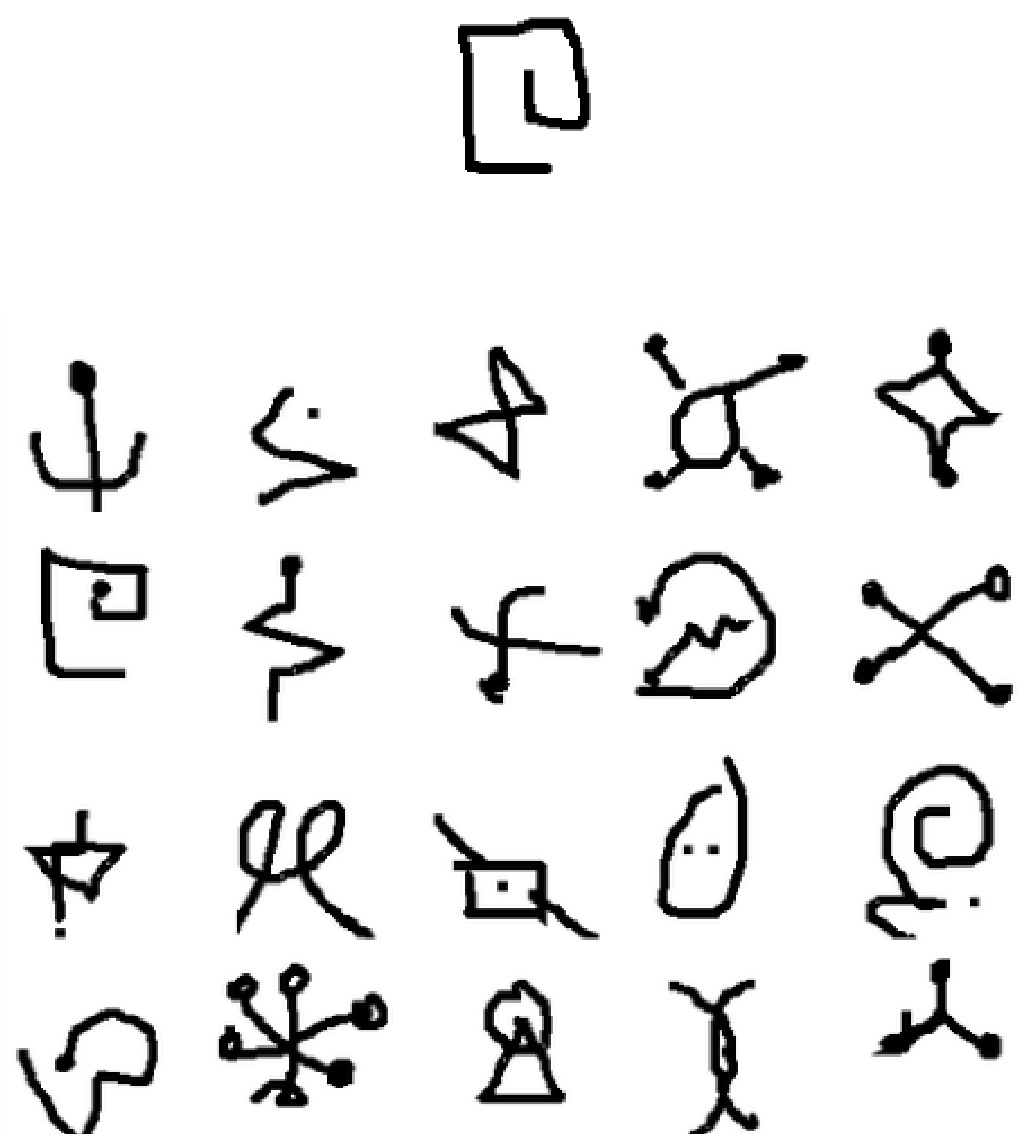
Omniglot

30 training alphabets
and 20 testing alphabets

Find match in 20 chars
from same alphabet
(chance is 95% error)

Network: 3 conv layers,
weighted L1 distance

Learnet predicts conv2



Omniglot

	Error (%)
Siamese	41.8
Siamese (unshared)	34.6
Learnet	28.6
Siamese learnet	31.4

Object tracking

Same problem as FC Siamese paper

Network: slim AlexNet (less channels) for speed

Learnet predicts conv2

Object tracking

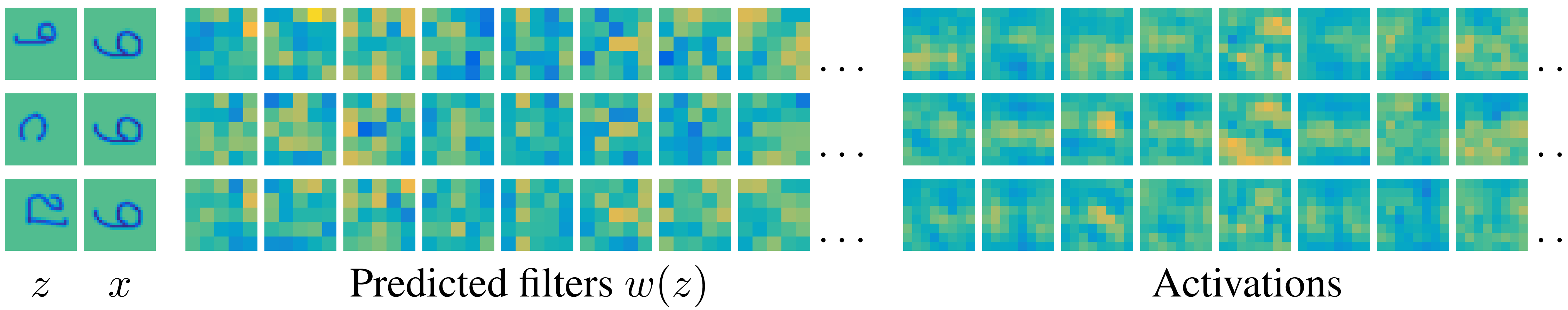
	Accuracy (IoU)	Num failures
Siamese	0.465	105
Siamese (unshared)	0.447	131
Siamese learnet	**0.500**	**87**

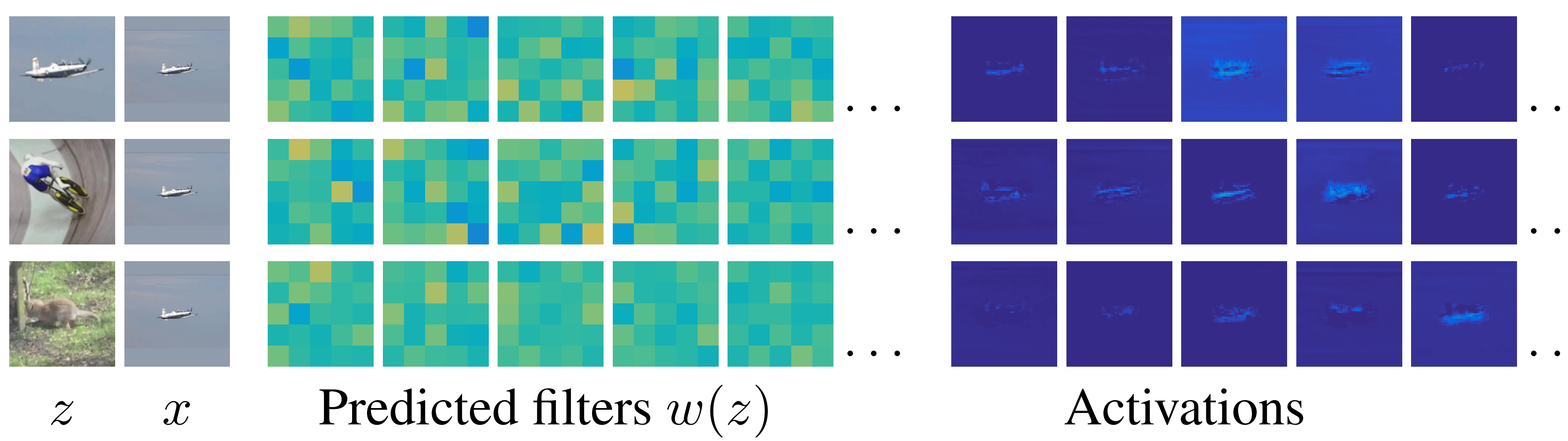
Object tracking

	Accuracy (IoU)	Num failures
Siamese	0.465	105
Siamese (unshared)	0.447	131
Siamese learnet	**0.500**	**87**

	Accuracy (IoU)	Num failures
DSST	0.483	163
MEEM	0.458	**107**
MUSTer	0.471	132
DAT	0.442	113
SO-DLT	**0.540**	108

Predicted filters





Conclusion

Learnets are an intriguing generalisation of siamese networks for one-shot learning

Much more to explore…

Predict filters at multiple layers

Alternative methods of reducing number of parameters (e.g. block-diag, sparse, hashing, …)

Structured ranking loss for Omniglot

The end

Thanks!

Questions?

Also feel free to contact us:
{luca.bertinetto, jack.valmadre}@eng.ox.ac.uk