Deep learning summer camp
3 Convolutional neural networks

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

July 4, 2016

Objectives of talk

- Regularisation
- Convolutional neural networks and
- what they can be used for.

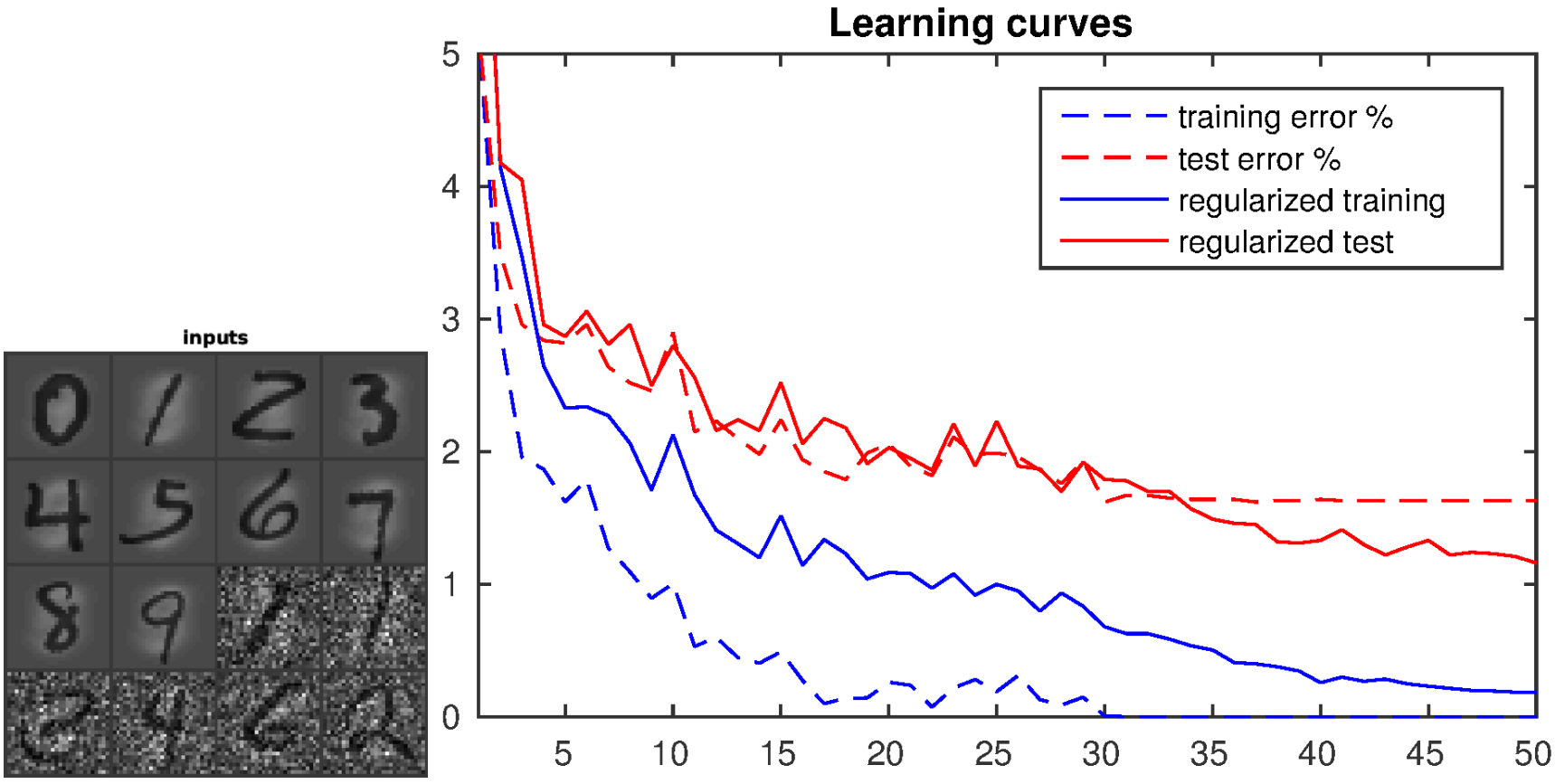
Part 1:

Tricks of the trade II
Regularisation

Goal of Regularization

- Neural networks are very powerful (universal appr.).
- Easy to perform great on the training set (overfitting).
- **Regularization** improves generalization to new data at the expense of increased training error.
- Use held-out validation data to choose hyperparameters (e.g. regularization strength).
- Use held-out test data to evaluate performance.

Example

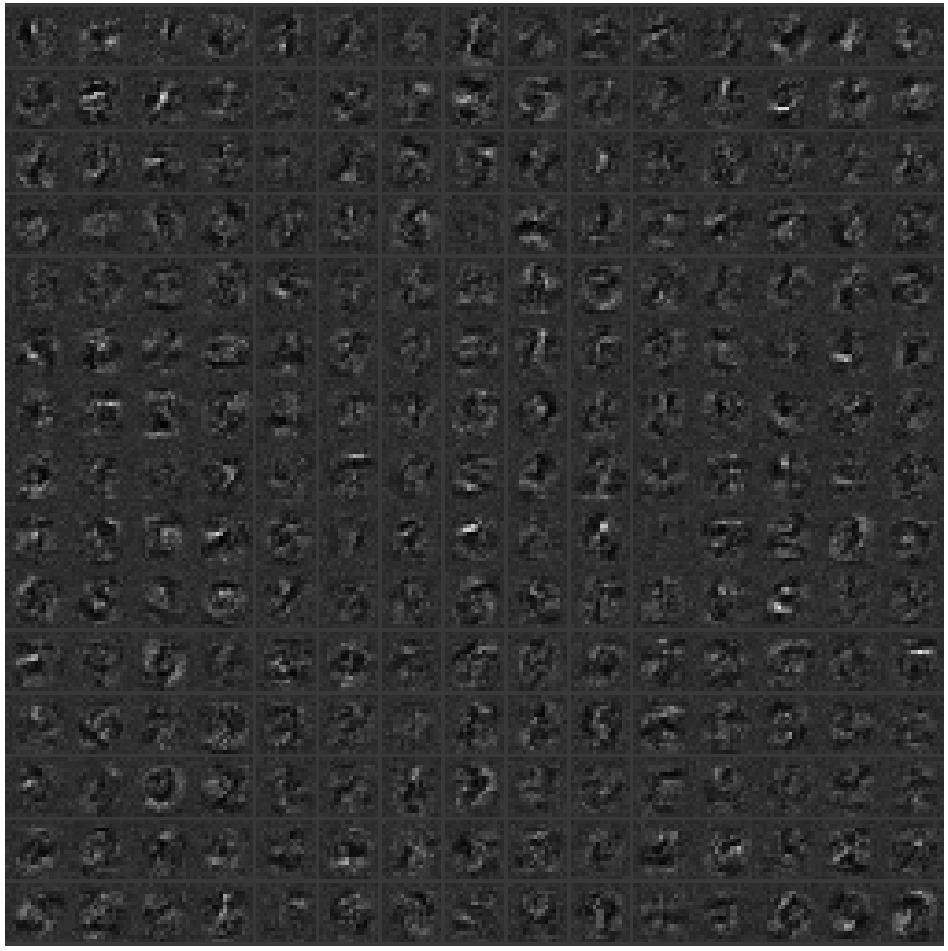


Without regularization training error goes to zero and learning stops.

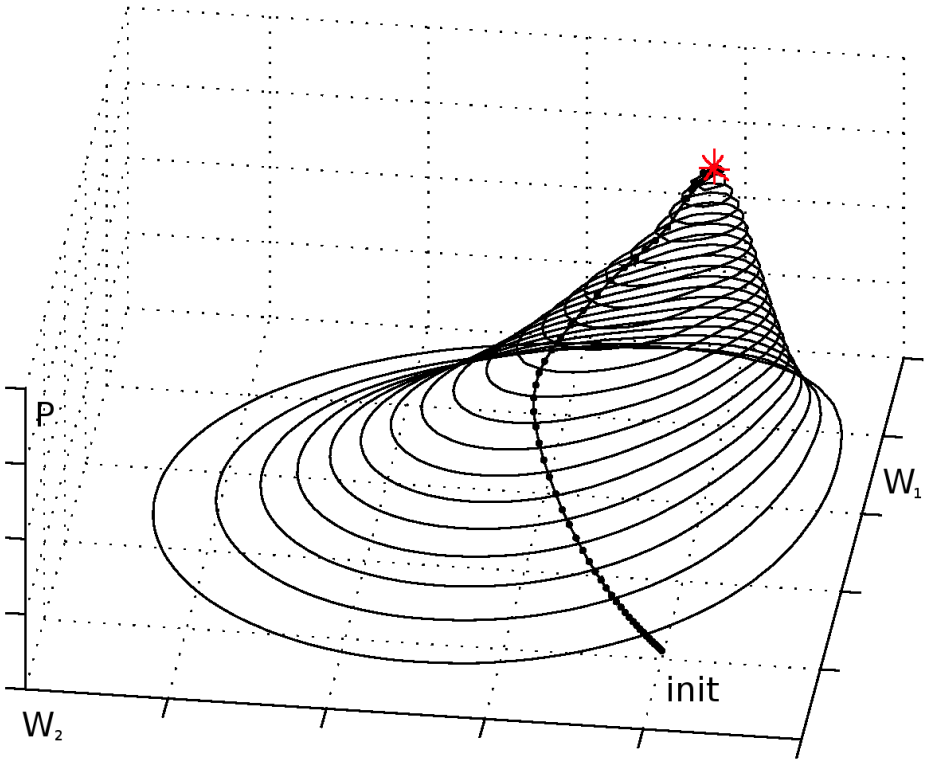
With noise regularization, test error keeps dropping.

Expressivity demo: Training first layer only

No regularization, training $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ only.
0.2% error on training set, 2% error on test set.



What is overfitting?



Posterior probability mass matters

Center of gravity $\neq$ maximum

Probability theory states how we should make predictions (of $\mathbf{y}_{\text{test}}$) using a model with unknowns $\boldsymbol{\theta}$ and data $\mathbf{X} = \{\mathbf{x}_{\text{train}}, \mathbf{y}_{\text{train}}, \mathbf{x}_{\text{test}}\}$:

$$P(\mathbf{y}_{\text{test}} \mid \mathbf{X}) = \int P(\mathbf{y}_{\text{test}}, \boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}$$

$$= \int P(\mathbf{y}_{\text{test}} \mid \boldsymbol{\theta}, \mathbf{X}) P(\boldsymbol{\theta} \mid \mathbf{X}) \mathrm{d}\boldsymbol{\theta}.$$

Probability of observing $\mathbf{y}_{\text{test}}$ can be acquired by summing or integrating over all different explanations $\boldsymbol{\theta}$. The term $P(\mathbf{y}_{\text{test}}|\boldsymbol{\theta}, \mathbf{X})$ is the probability of $\mathbf{y}_{\text{test}}$ given a particular explanation $\boldsymbol{\theta}$ and it is weighted with the probability of the explanation $P(\boldsymbol{\theta}|\mathbf{X})$. However, such computation is intractable. If we want to choose a single $\boldsymbol{\theta}$ to represent all the probability mass, it is better not to overfit to the highest probability peak, but to find a good representative of the mass.

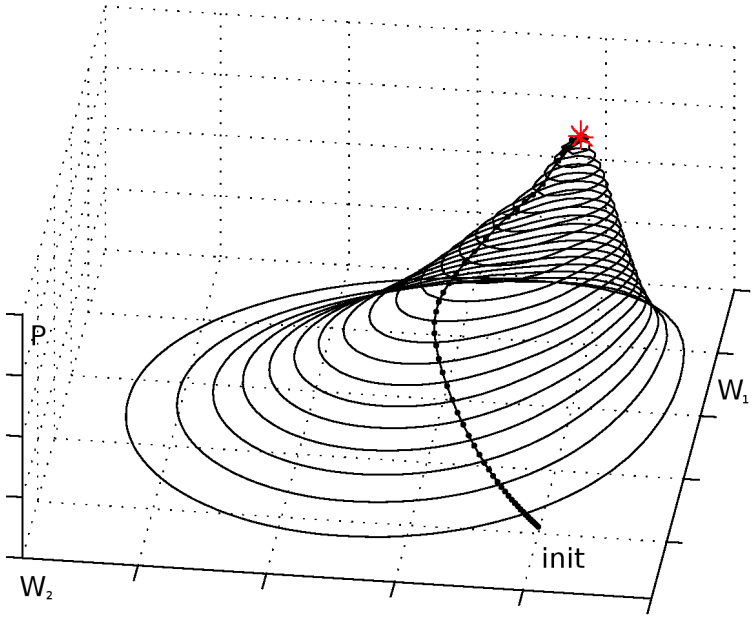
Regularization methods

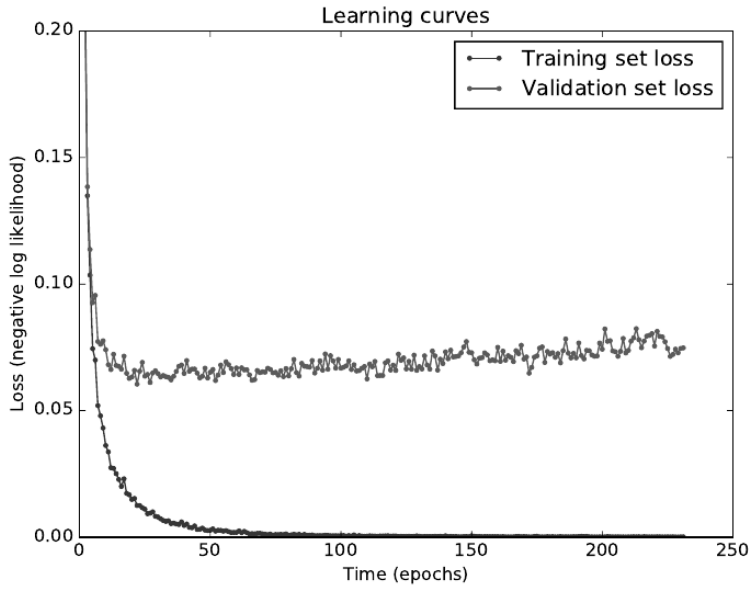
- Limited size of network
- Early stopping
- Weight decay
- Data augmentation
- Injecting noise
- Parameter sharing (e.g. convolutional)
- Sparse representations
- Ensemble methods
- Auxiliary tasks (e.g. unsupervised)
- Probabilistic treatment (e.g. variational methods)
- Adversarial training, . . .

Limited size of network

- Rule of thumb:
  When #parameters is ten times less than #outputs $\times$ #examples, overfitting will not be severe.
- Reducing input dimensionality (e.g. by PCA) helps in reducing parameters
- Easy. Low computational complexity
- Other methods give better accuracy
- *Data augmentation* increases #examples
  Parameter sharing decreases #parameters
  Auxiliary tasks increases #outputs

Early stopping



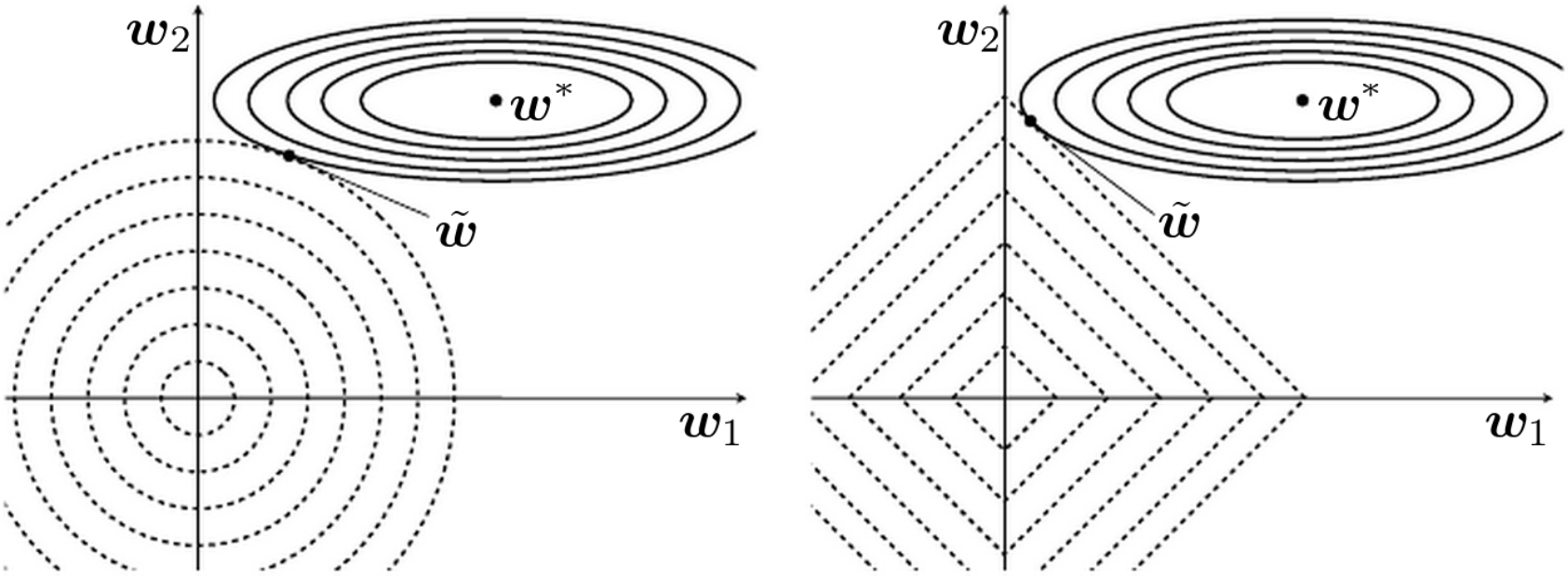


- Monitor validation performance during training
- Stop when it starts to deteriorate
- With other regularization, it might never start
- Keeps solution close to the initialization

Weight decay (Tikhonov, 1943)

- Add a penalty term to the training cost $C = \cdots + \Omega(\boldsymbol{\theta})$
  Note: only a function of parameters $\boldsymbol{\theta}$, not data.
- L^2 regularization: $\Omega(\boldsymbol{\theta}) = \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2$
  hyperparameter λ for strength.
  Gradient: $\frac{\partial\Omega(\boldsymbol{\theta})}{\partial\theta_i} = \lambda\theta_i$.
- L^1 regularization: $\Omega(\boldsymbol{\theta}) = \lambda/2\|\boldsymbol{\theta}\|_1$
  Gradient: $\frac{\partial\Omega(\boldsymbol{\theta})}{\partial\theta_i} = \lambda\text{sign}(\theta_i)$.
  Induces sparsity: Often many params become zero.
- Max-norm: Constrain row vectors $\mathbf{w}_i$ of weight matrices to $\|\mathbf{w}_i\|^2 \leq c$.

Weight decay



- L2 (left) and L1 (right).
- w^* unregularised solution, $\tilde{w}$ regularised solution.
- Note: L1 pushes small parameters more towards zero - sparsity!

Weight decay as Bayesian prior

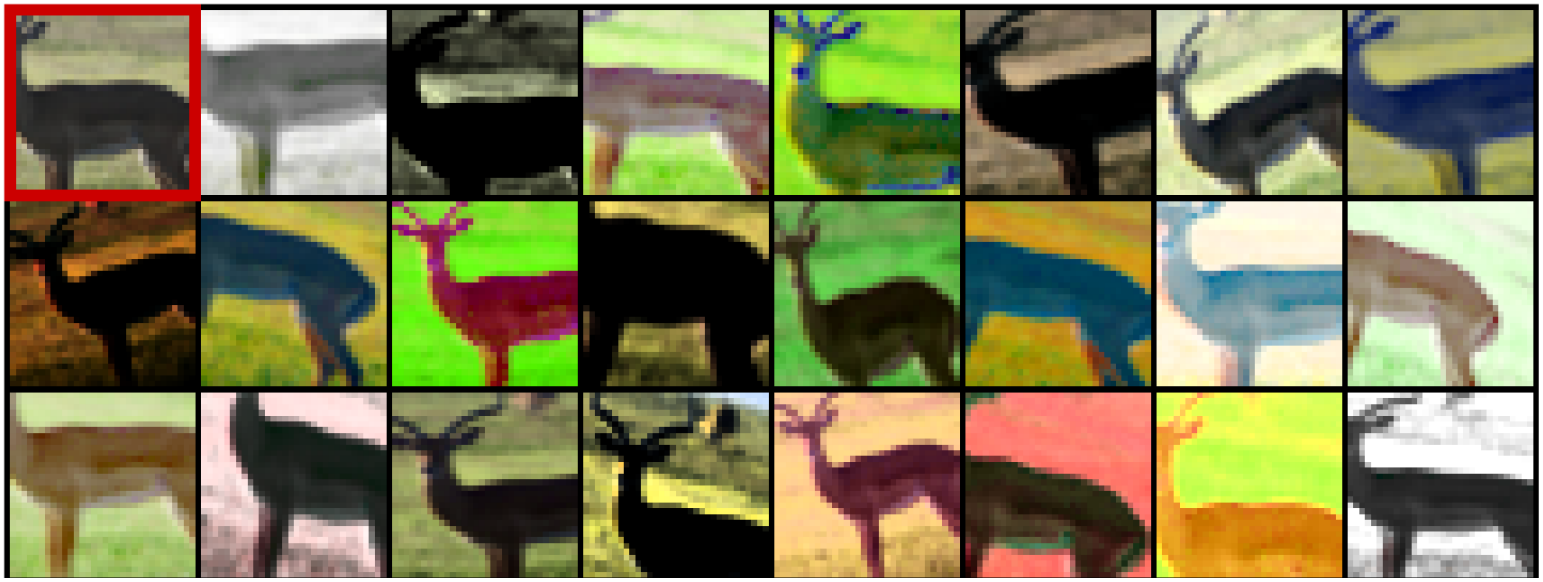
- Consider the maximum a posteriori solution
- Bayes rule: $P(\theta \mid \mathbf{X}) = P(\mathbf{X}|\theta)P(\theta)$
  written on -log scale: $C = -\log P(\mathbf{X}|\theta) - \log P(\theta)$
- Assuming Gaussian prior $P(\theta) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$
  we get $\Omega(\theta) = \sum_i -\log \exp \frac{-\theta_i^2}{2\lambda^{-1}} = \frac{\lambda}{2}\|\theta\|^2$
- L^2 regularization $\Leftrightarrow$ Gaussian prior
- L^1 regularization $\Leftrightarrow$ Laplace prior
- Max-norm regularization $\Leftrightarrow$ Uniform prior with finite support
- $\Omega = 0 \Leftrightarrow$ Maximum likelihood

How to set hyperparameter λ?

- In general, difficult to set strength of regularization
- Split data into training, validation, and test sets
- Choose a number of settings λ, train separately
- Use validation performance to pick the best λ
- (Retrain using both training and validation sets)
- Evaluate final performance on test data
- Ongoing work on adjusting hyperparameters on the fly (Luketina et al., 2016)

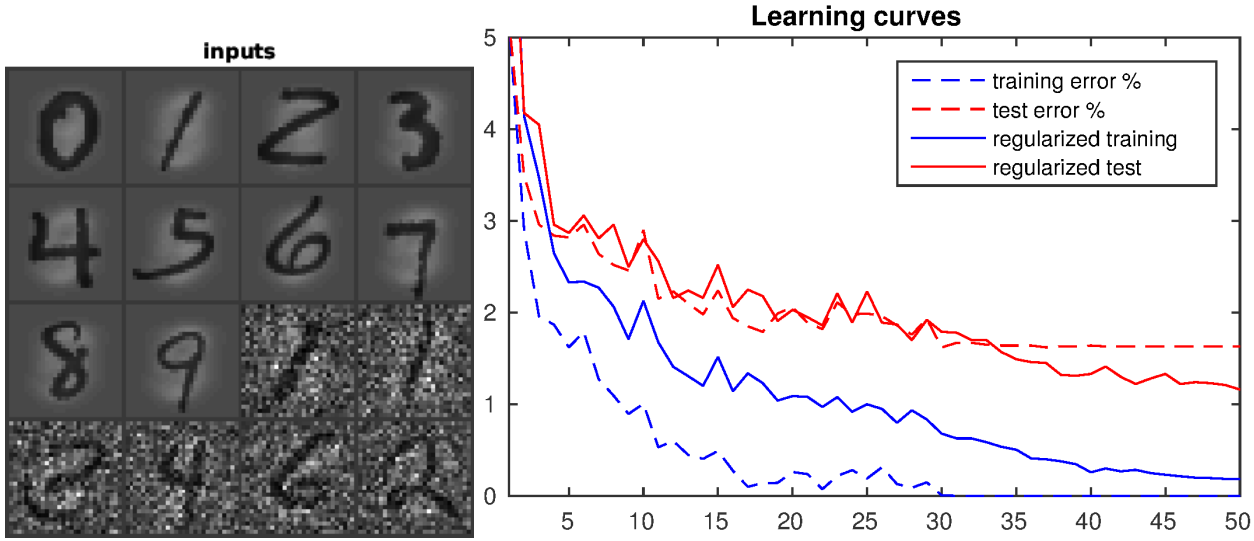
Data augmentation

Image from (Dosovitskiy et al., 2014)



Augmented data by image-specific transformations.
E.g. cropping just 2 pixels gets you 9 times the data! Infinite MNIST: `http://leon.bottou.org/projects/infimnist`

Injecting noise (Sietsma and Dow, 1991)



- Inject random noise during training
  separately in each epoch
- Can be applied to input data, to hidden activations, or to weights
- Can be seen as data augmentation
- Simple end effective

Injecting noise to inputs (analysis)

- Inject small additive Gaussian noise at inputs
- Assume least squares error at output **y**
- Taylor series expansion around **x**
- $\Rightarrow$ Corresponds to penalizing the Jacobian $\|\mathbf{J}\|^2$

$$\mathbf{J} = \frac{d\mathbf{y}}{d\mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_c}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_d} & \cdots & \frac{\partial y_c}{\partial x_d} \end{pmatrix}$$

- For linear networks, this reduces to L^2 penalty
- Rifai et al. (2011) penalize the Jacobian directly

Parameter sharing

- Force sets of parameters to be equal
- Reduces the number of (unique) parameters
- Important in convolutional networks (CNNs, this lecture)
- Auto-encoders sometimes share weights between encoder and decoder (Unsupervised learning lecture)

Sparse representations

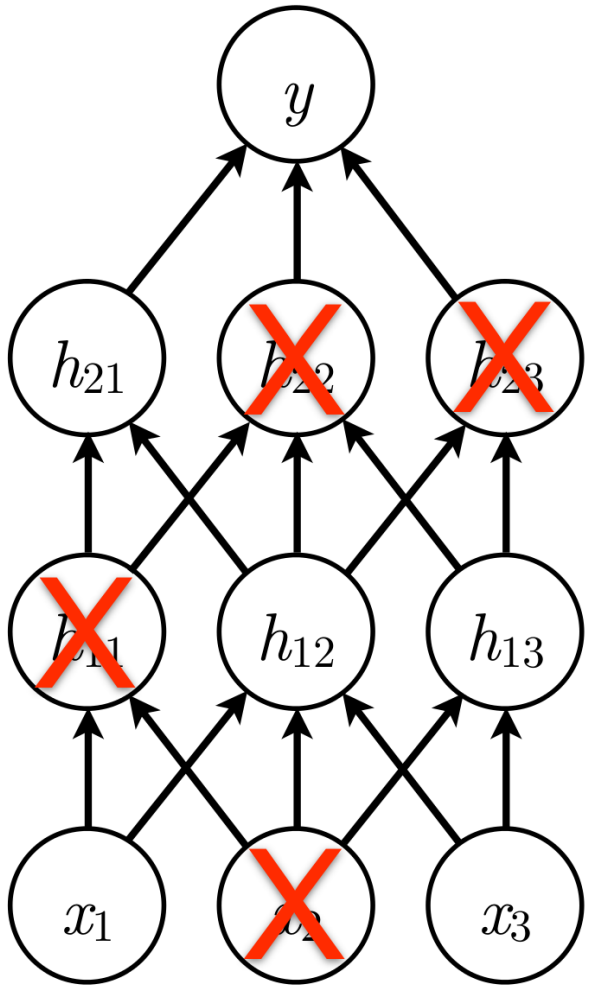
- Penalize representation $\mathbf{h}$ using $\Omega(\mathbf{h})$ to make it sparse
- L^1 penalty on weights makes $\mathbf{W}$ sparse
- Similarly L^1 penalty can make $\mathbf{h}$ sparse
- Also possible to set a desired sparsity level
- *Sparse coding* is common in image processing

Ensemble methods

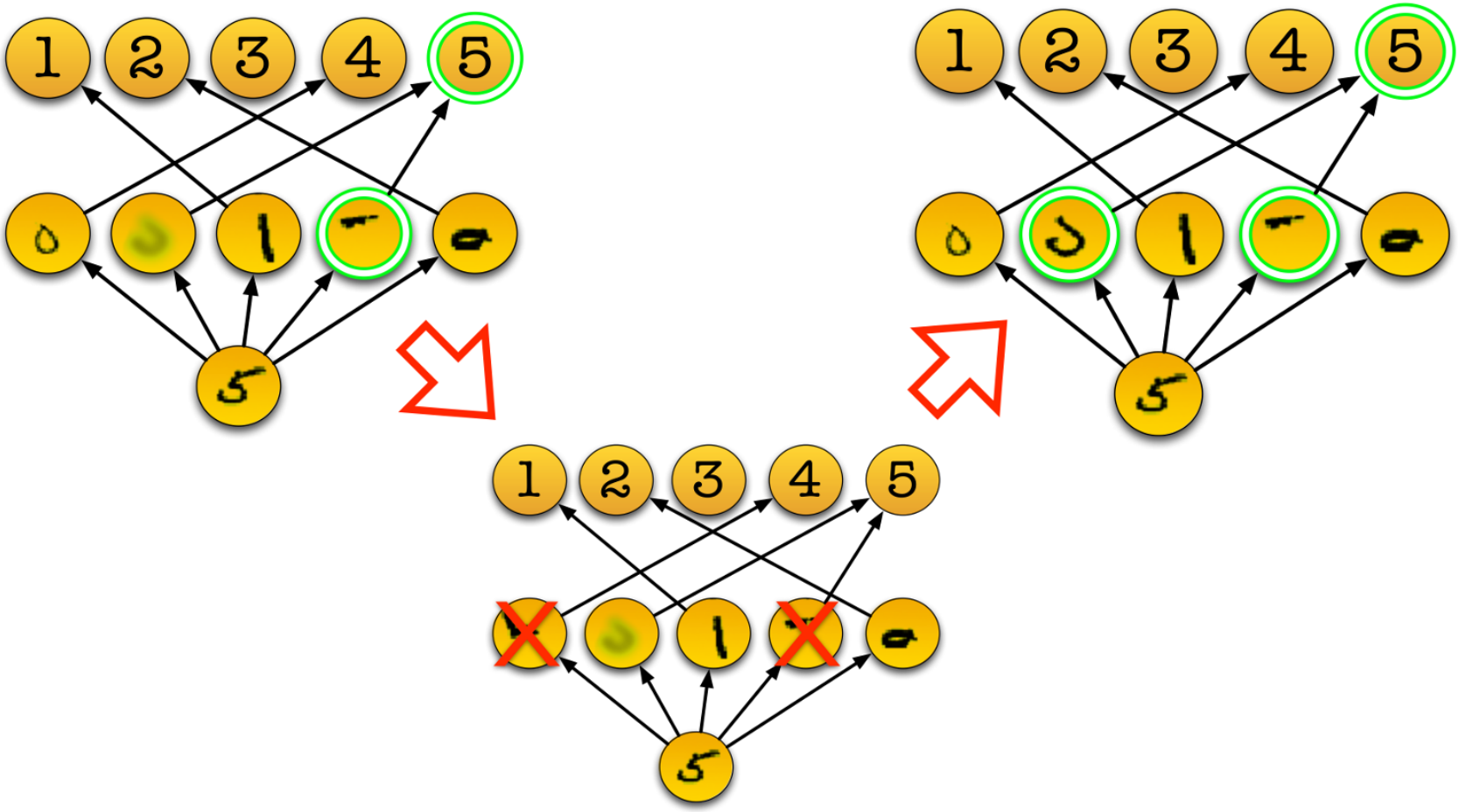
- Train several models and take average of their outputs
  Instead of one point representing $P(\theta|\mathbf{X})$, use several
- Also known as *bagging* or *model averaging*
- It helps to make individual models different by
  - varying models or algorithms
  - varying hyperparameters
  - varying data (dropping examples or dimensions)
  - varying random seed
- It is possible to train a single final model to mimick the performance of the ensemble, for test-time computational efficiency (Hinton et al., 2015)

Dropout (Srivastava et al., 2014)

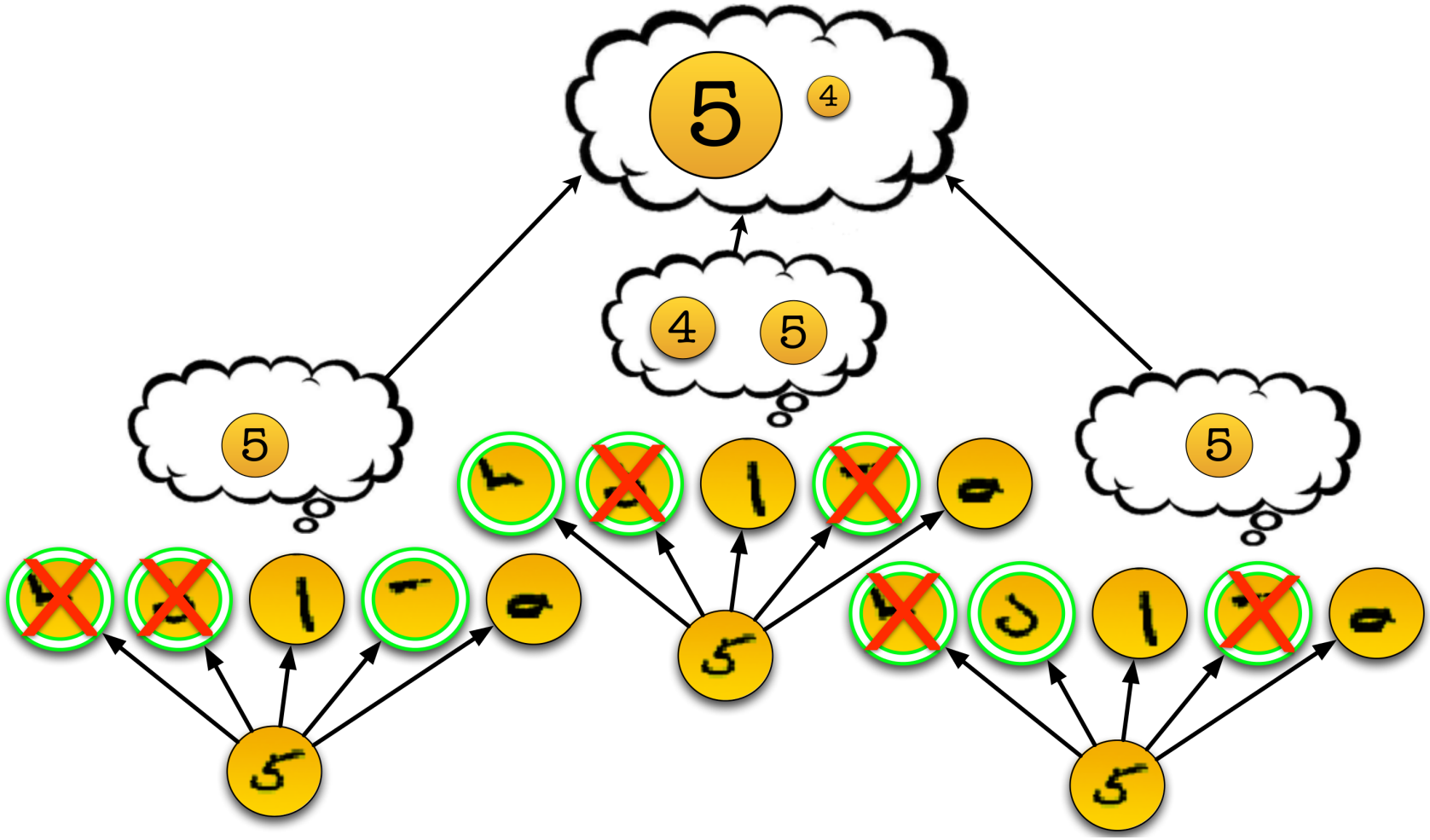
- Each time we present data example **x**, randomly delete each hidden node with 0.5 probability
- Can be seen as *injecting noise* or as *ensemble*:
  - Multiplicative binary noise
  - Training an ensemble of $2^{|h|}$ networks with weight sharing
- At test time, use all nodes but divide weights by 2



Dropout training



Dropout as bagging



Auxiliary tasks

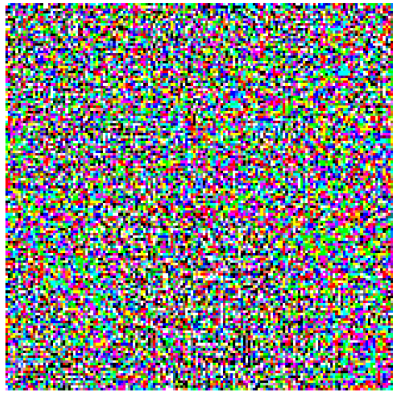
- Multi-task learning: *Parameter sharing* between multiple tasks
- E.g. speech recognition and speaker identification could share low-level representations
- Layer-wise pretraining (Hinton and Salakhutdinov, 2006) can be seen as using unsupervised learning as an auxiliary task

Probabilistic treatment

- Probabilistic modelling is strong but complex form of regularization
- Some keywords:
  - Variational methods: E.g. model means and variances of signals
  - Sampling = Markov chain Monte Carlo (MCMC)
  - Boltzmann machines

Adversarial training (Szegedy et al., 2014)

$$+ .007 \times$$



$$=$$

$\boldsymbol{x}$

$y =$ “panda”

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

$\boldsymbol{x} +$
$\epsilon \, \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
“gibbon”

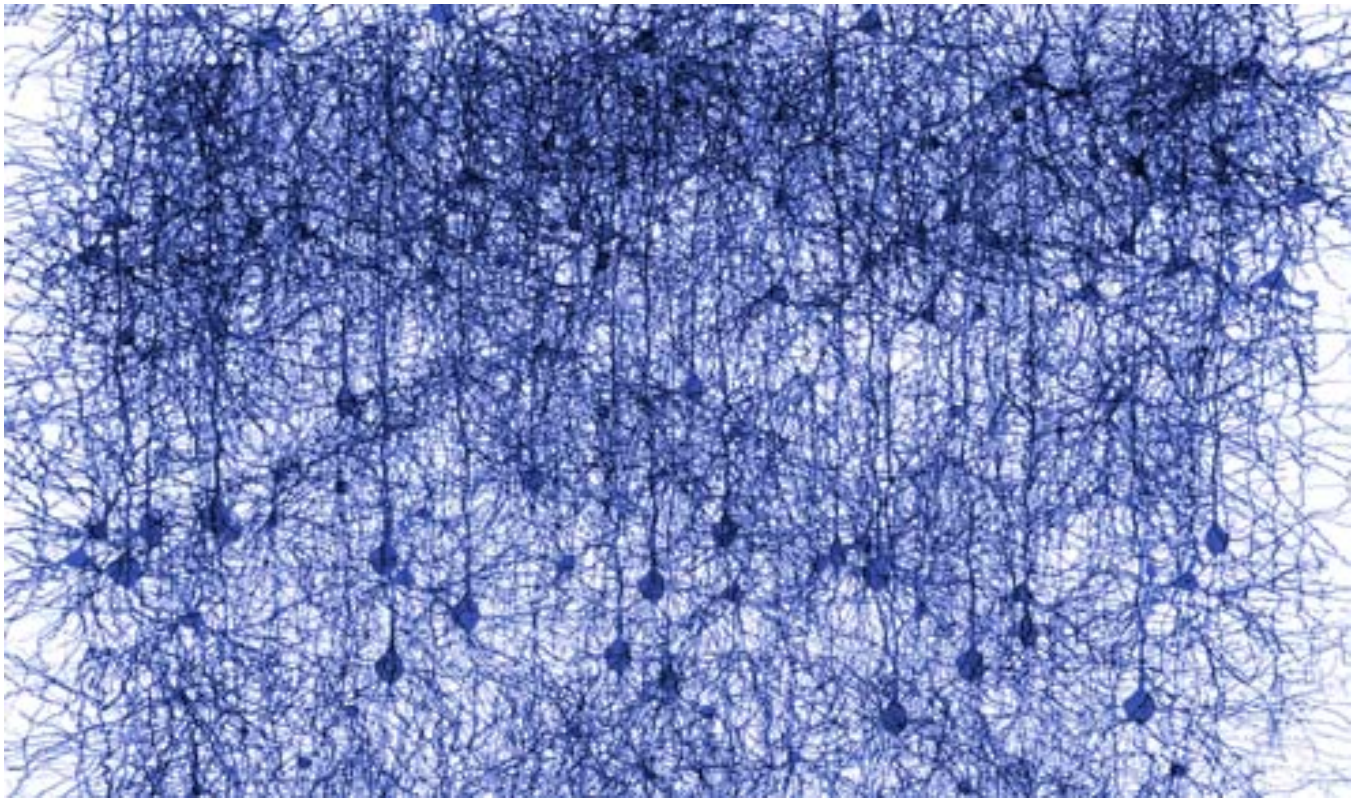
- Search for an input **x**′ near a datapoint **x** that would have very different output **y**′ from **y**
- Adversaries can be found surprisingly close!
- Miyato et al. (2016) build a very effective regulariser

Part 2:

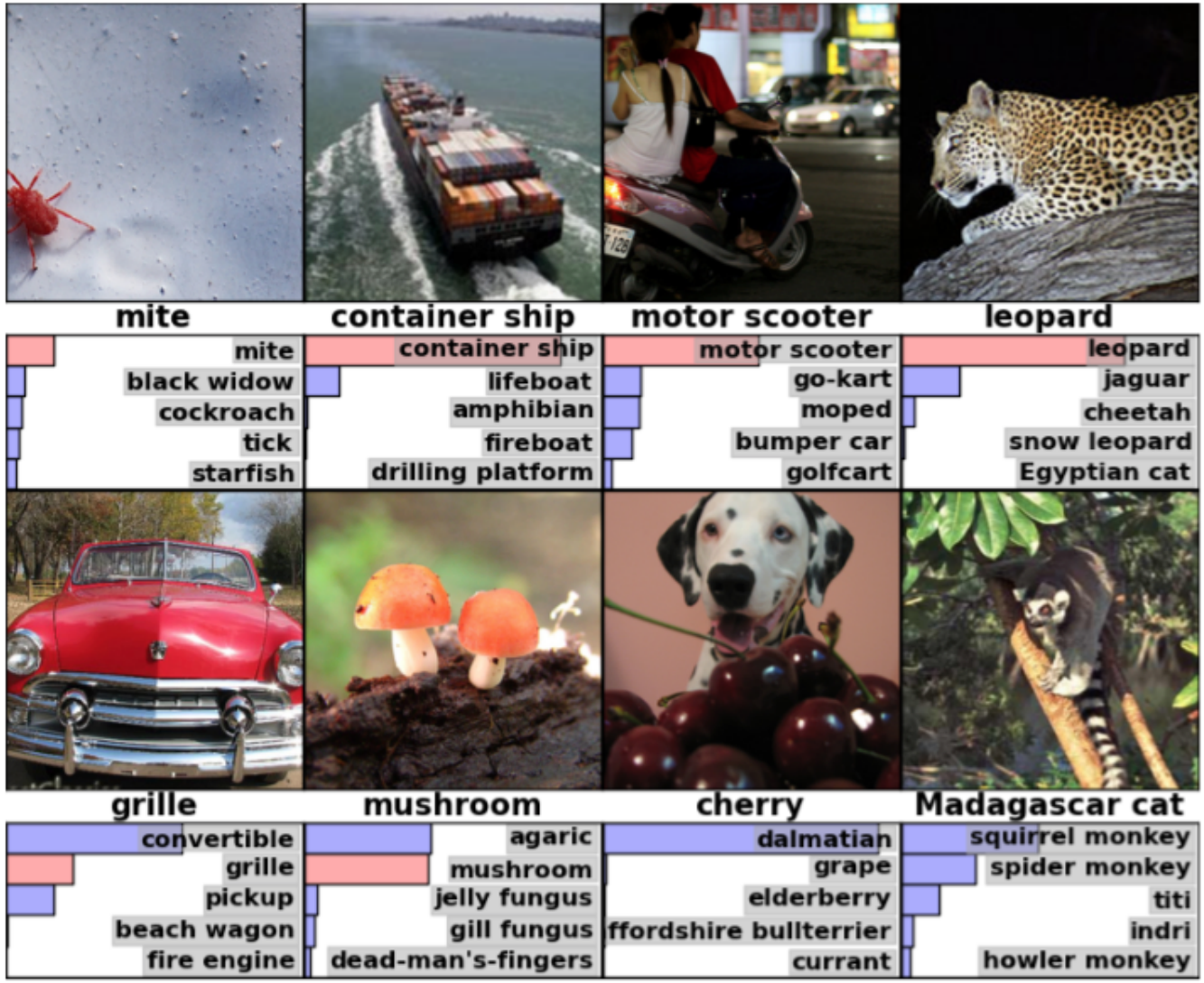
Convolutional NNs
Definition and uses

Neural networks (NNs)

- Feedforward neural networks (FFNNs)
- Convolutional neural networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Auto-encoders (AE)

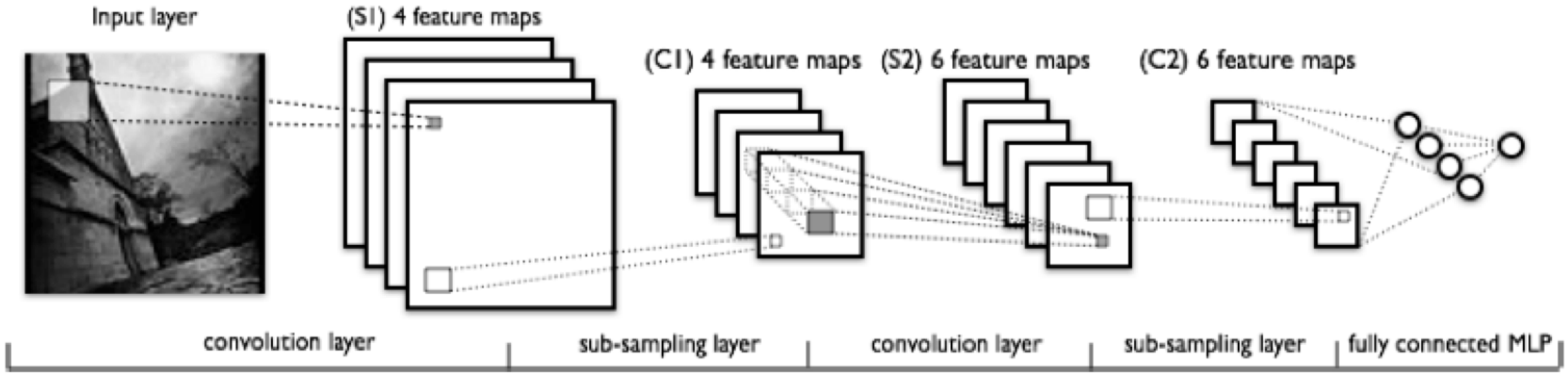


ImageNet - image classification



- 1.000 different classes - including many types of dogs!
- 1.000.000 training images

Convolutional neural networks



$$\begin{bmatrix} 10 & 0 & -10 \\ 0 & 0 & 0 \\ -10 & 0 & 10 \end{bmatrix}$$

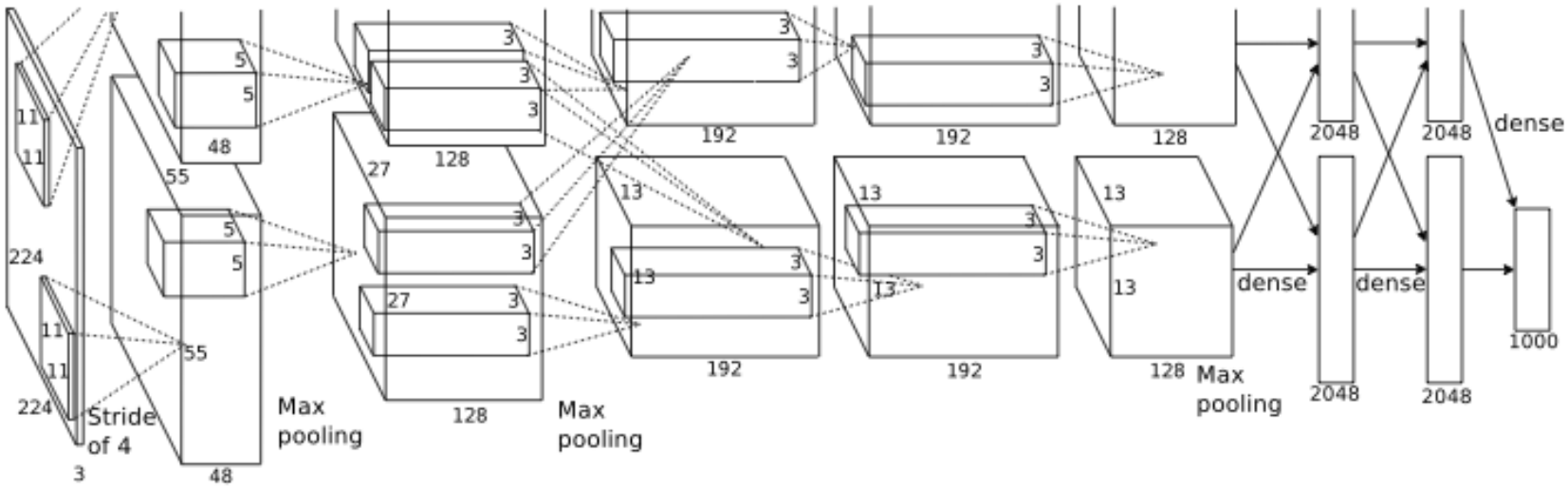
Feature engineering vs engineered models

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

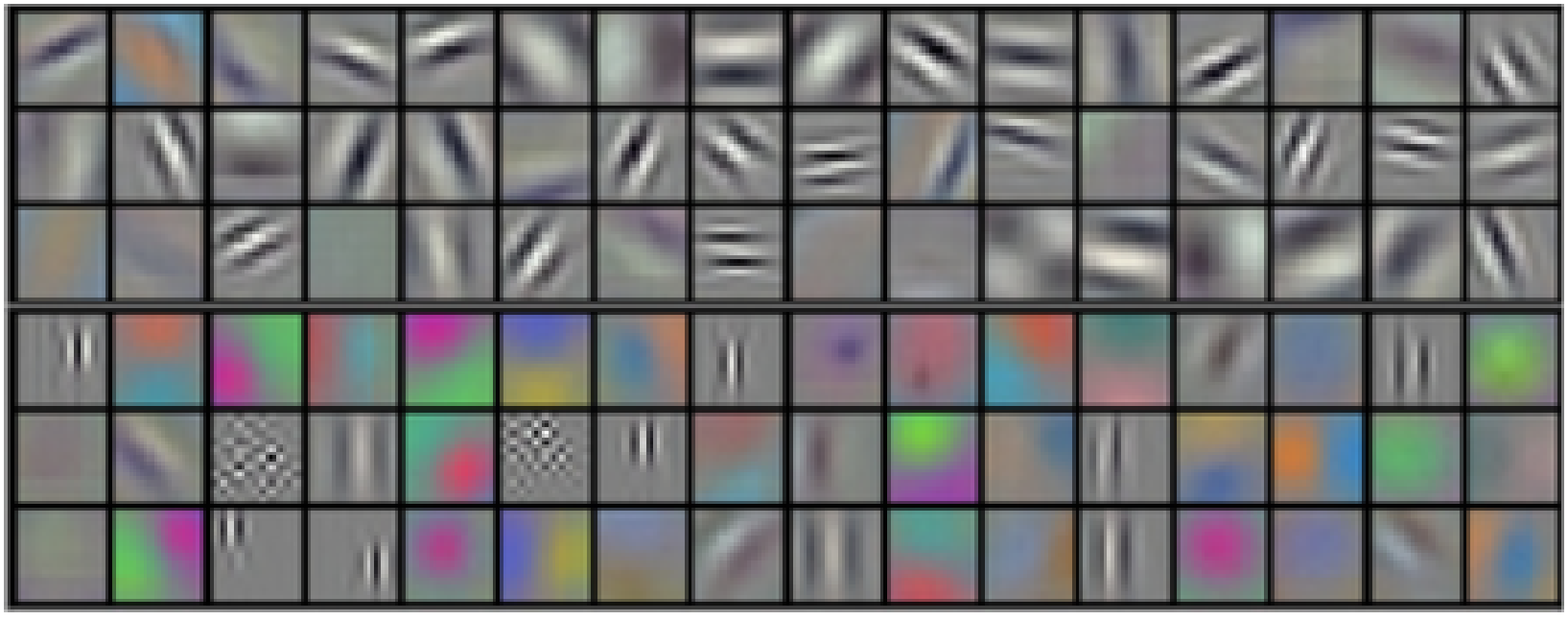
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



www.cs.toronto.edu/~fritz/absps/imagenet.pdf

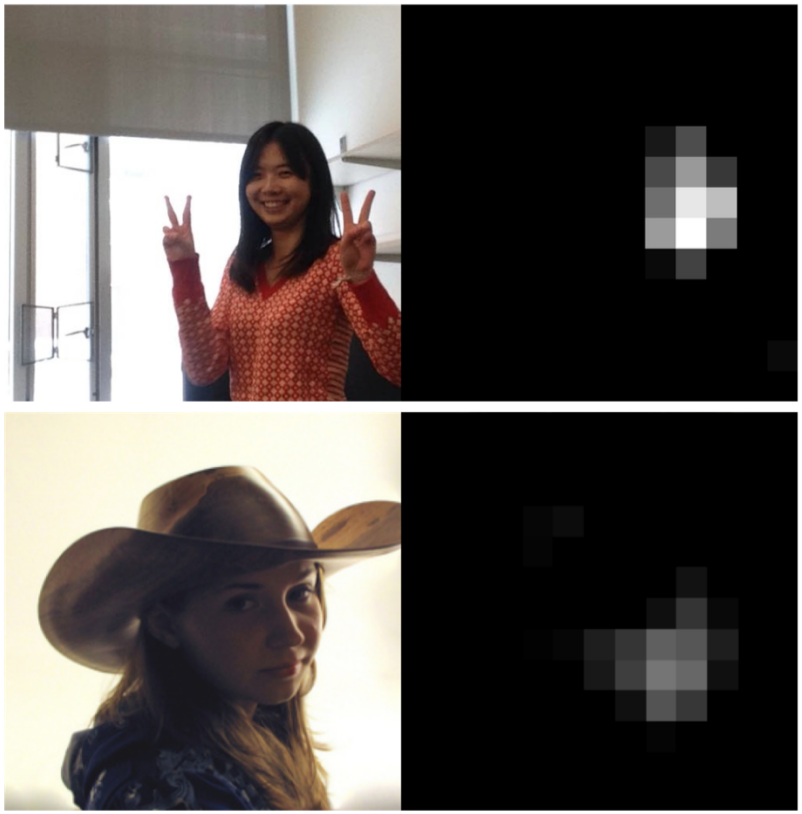
Learned filters in first layer



- Seminal paper:
- Krizhevsky et al, ImageNet Classification with Deep Convolutional Neural Networks
- `www.cs.toronto.edu/~fritz/absps/imagenet.pdf`

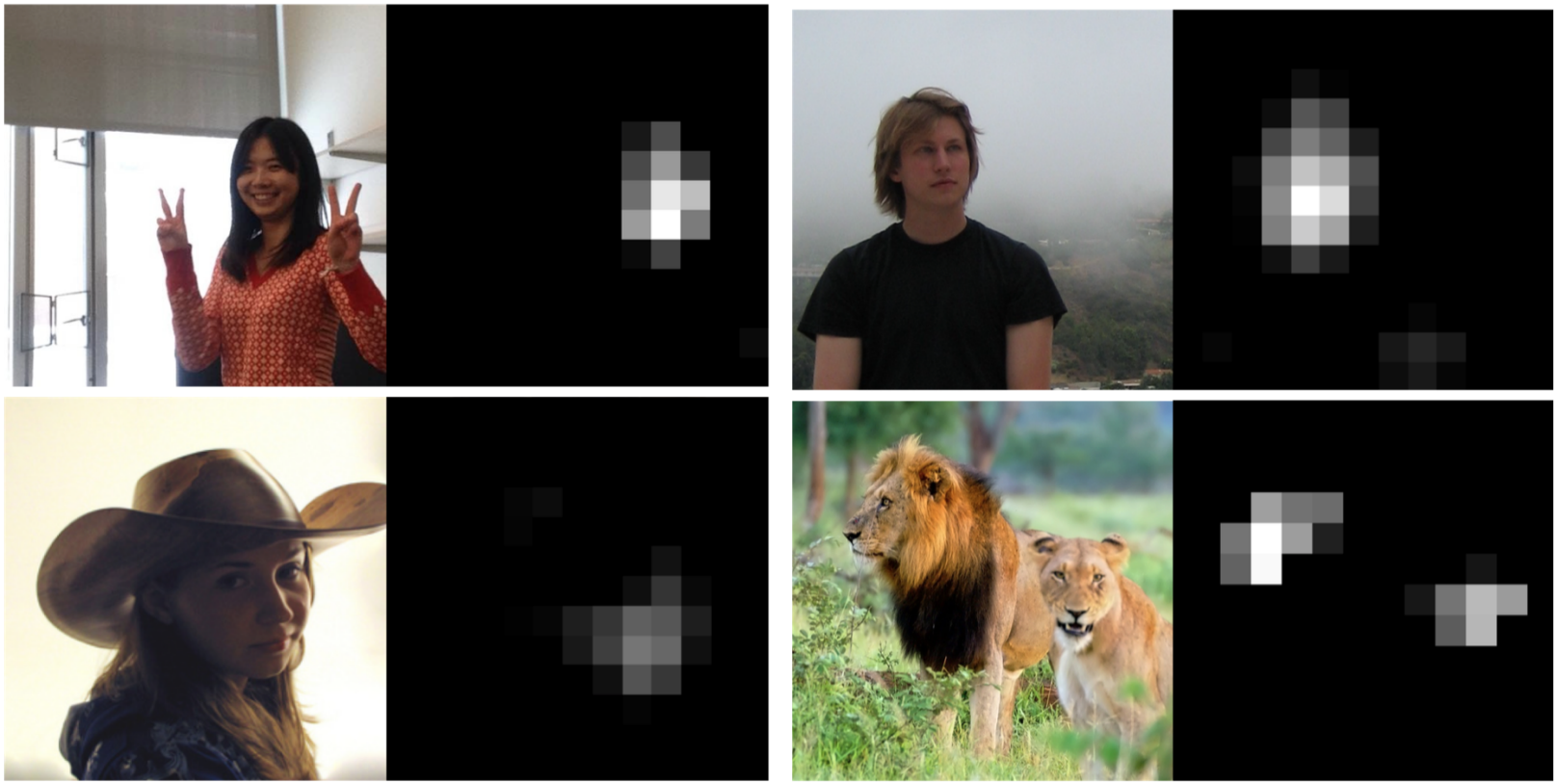
Emergent higher level abstractions

- Look at output of filter in 5th layer!



Emergent higher level abstractions

- Look at output of filter in 5th layer!



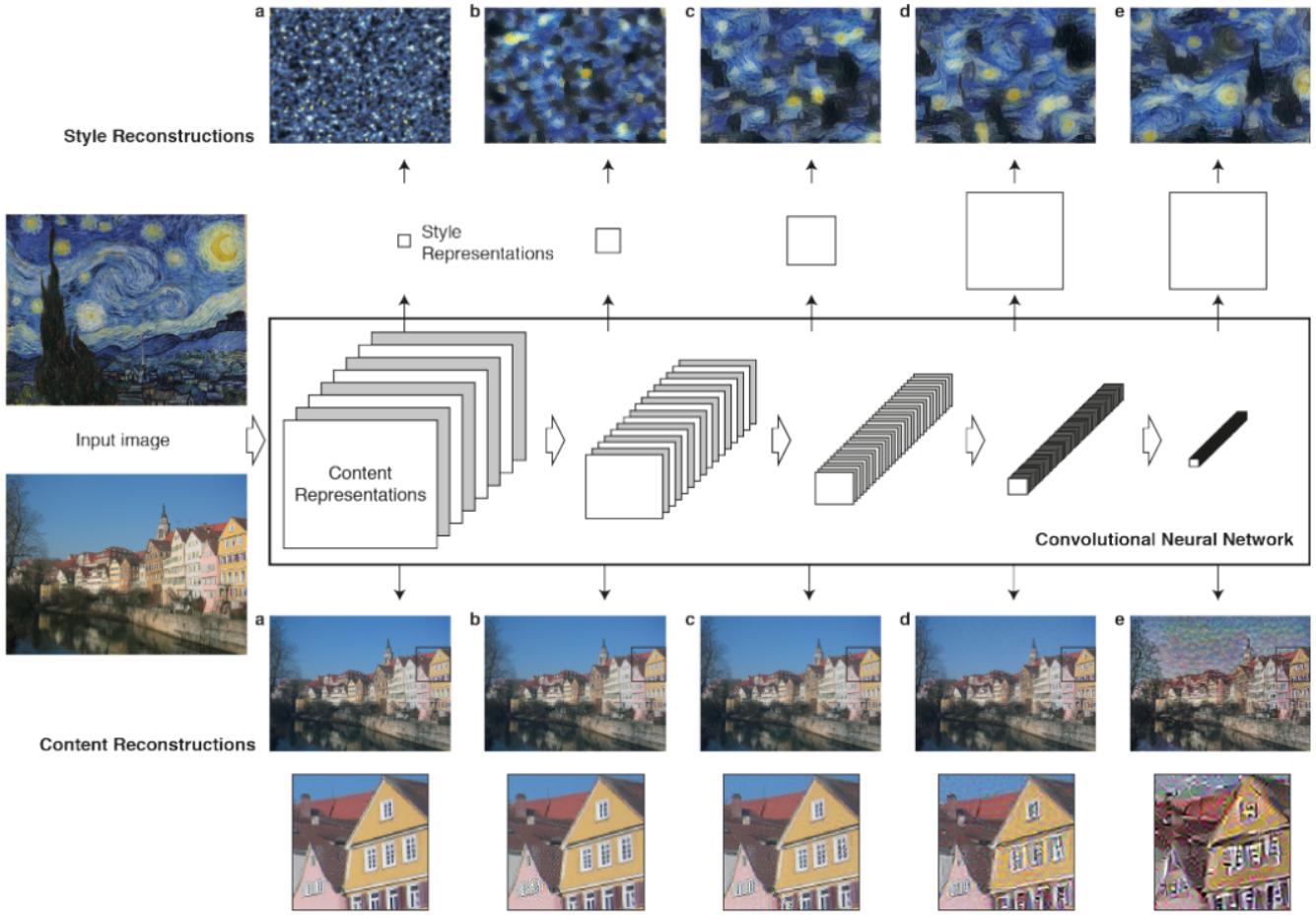
Yosinski et. al., ICML, google: deepvis

Google Deep Dream

http://deepdreamgenerator.com/

A Neural Algorithm of Artistic Style

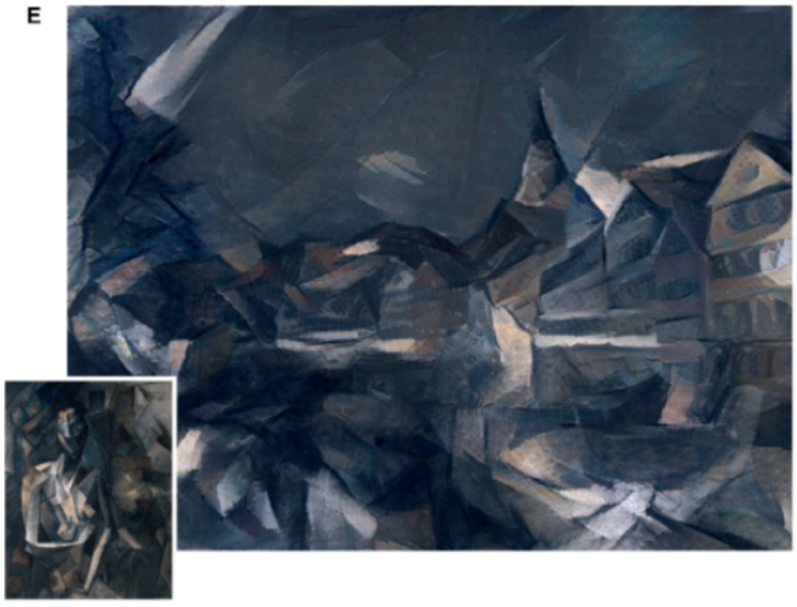
- Gatys, Ecker and Bethge
  `http://arxiv.org/pdf/1508.06576v2.pdf`
- Idea: Separate content and style



Borrowing the style from the masters!

Borrowing the style from the masters!

Borrowing the style from the masters!



Here should have been a live demo

DTU president

©Anders Boesen Lindbo Larsen

Neural artistic style - The Movie

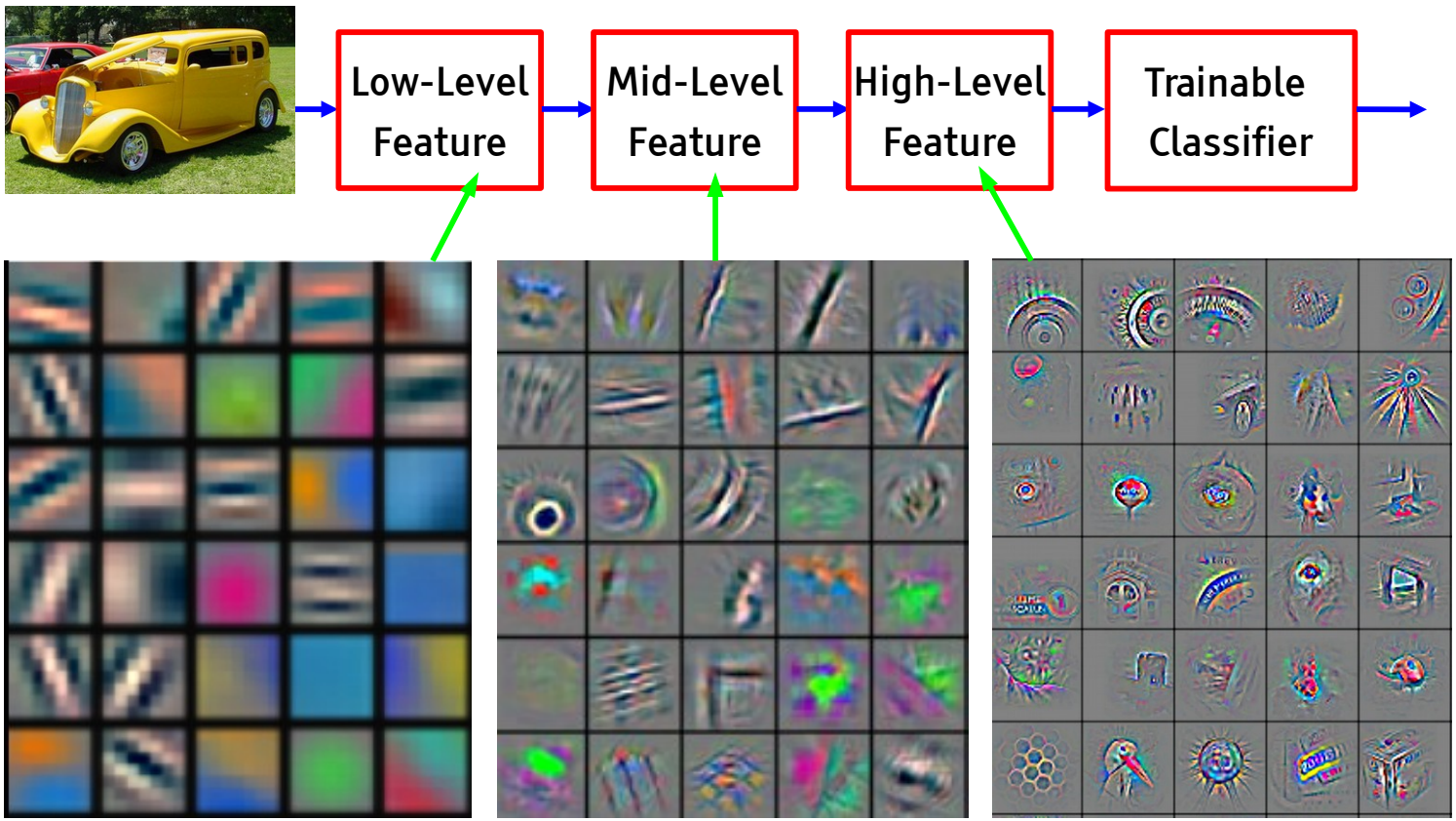
Part 3:

Convolutional NNs
The details

Deep Learning = Learning Hierarchical Representations

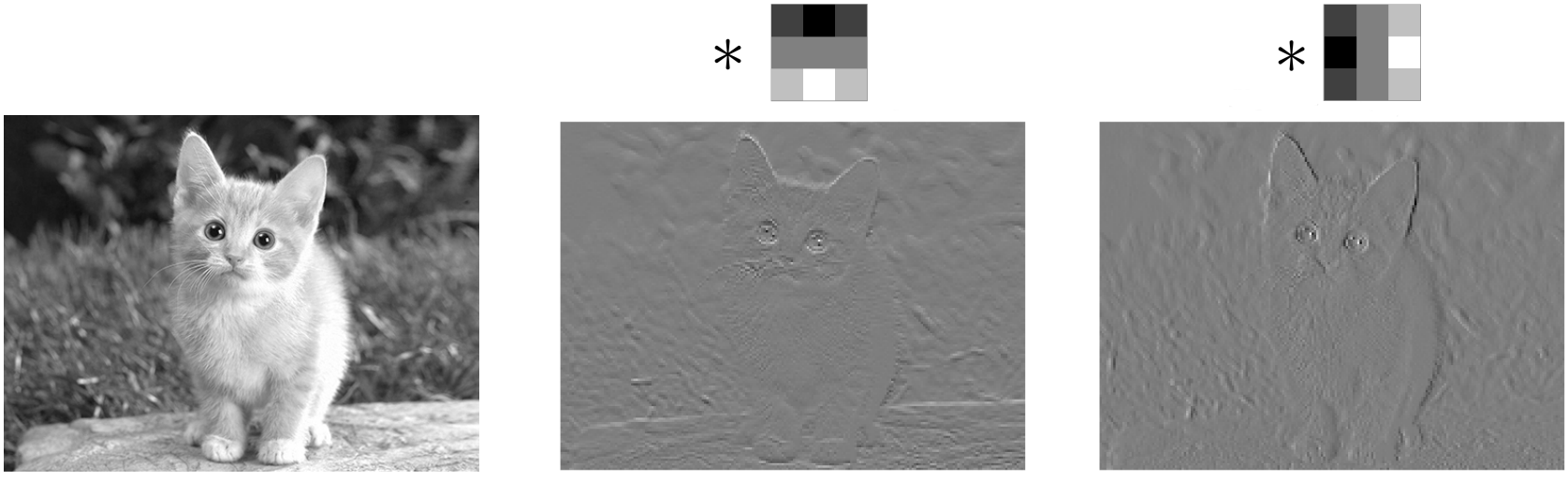
Y LeCun

- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolution $*$



$$(w * x)[t] = \sum_a w[a]x[t-a]$$

Convolutional Networks (LeCun et al., 1989)

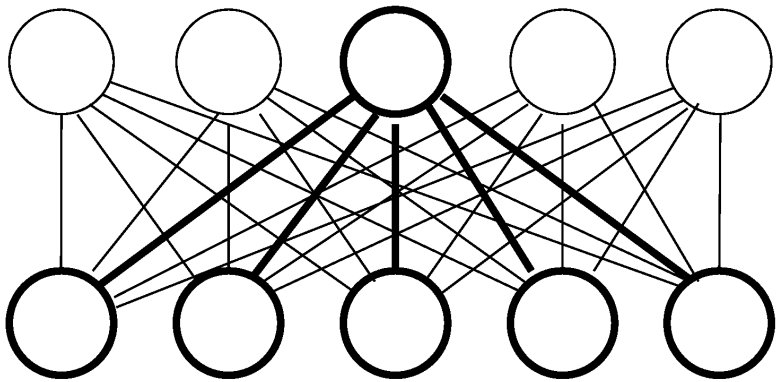
- How would MNIST intro network scale to real images?
- Instead of hundreds of pixels, a million.
- Would weight matrices $\mathbf{W}$ be million $\times$ million?

Convolutional Networks (LeCun et al., 1989)

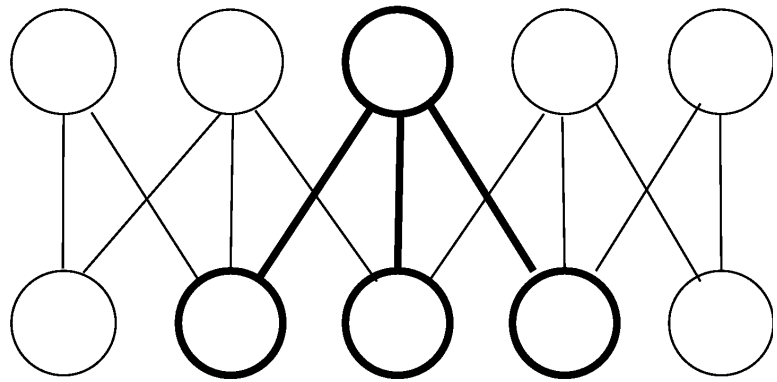
- How would MNIST intro network scale to real images?
- Instead of hundreds of pixels, a million.
- Would weight matrices $\mathbf{W}$ be million $\times$ million?

- Change notation: Replace vector signals with tensors indexed by x, y, c for x, y coordinates and channel c.
- Weight between $h^{(1)}_{x_1,y_1,c_1}$ and $h^{(2)}_{x_2,y_2,c_2}$ is $W^{(2)}_{x_1,x_2,y_1,y_2,c_1,c_2}$.
- Two ideas: Local connectivity and parameter sharing

Local connectivity



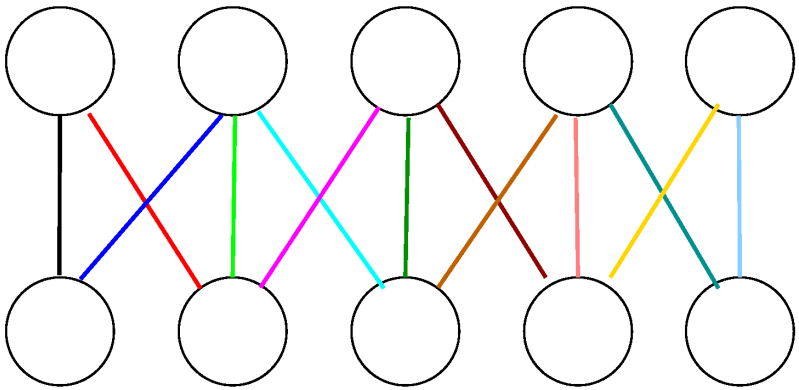
25 parameters



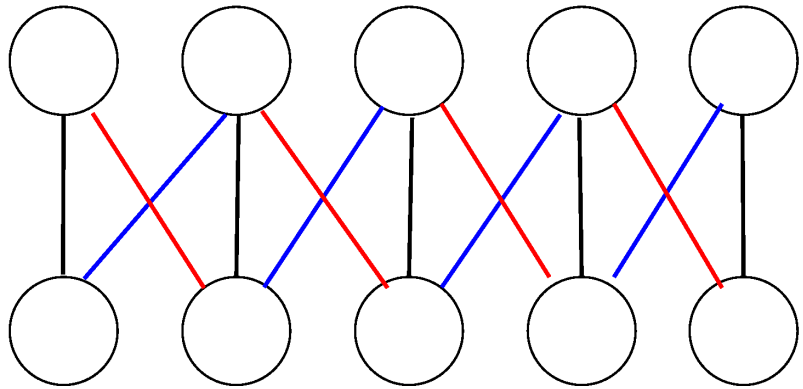
13 parameters

- Weight is zero unless $|x_1 - x_2| \leq k$ and $|y_1 - y_2| \leq k$

Parameter sharing



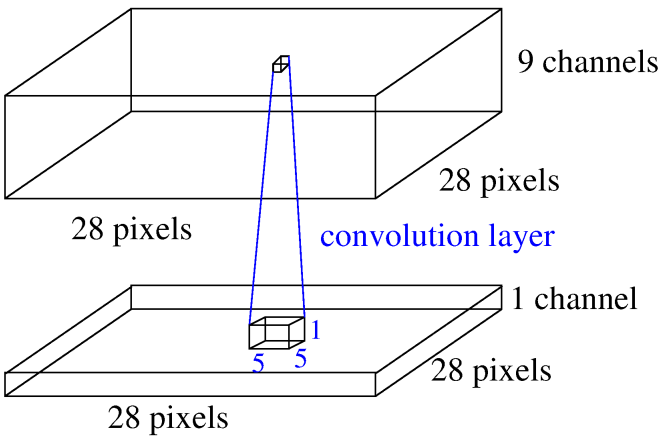
13 parameters

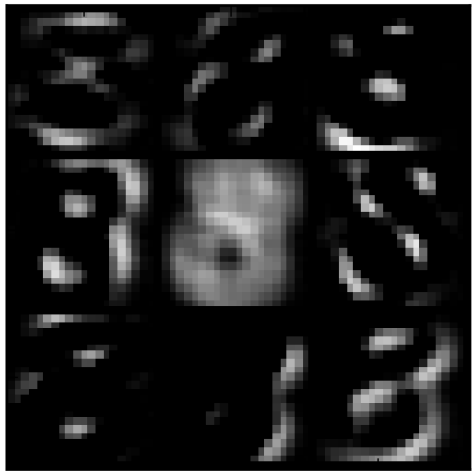


3 parameters

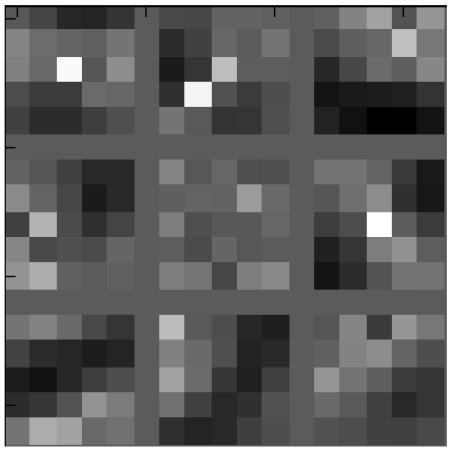
- Weights indexed by $\Delta x = x_1 - x_2$ and $\Delta y = y_1 - y_2$.
- $W_{\Delta x, \Delta y, c_1, c_2}$ replaces $W_{x_1, x_2, y_1, y_2, c_1, c_2}$.

Example (MNIST again)

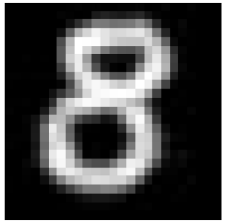




signals



weights



- MNIST data is 28×28 pixels and 1 channel.
- First hidden layer has 28×28 pixels and 9 channels.
- Use 5×5 filters ($\Delta x \leq 2$ and $\Delta y \leq 2$).

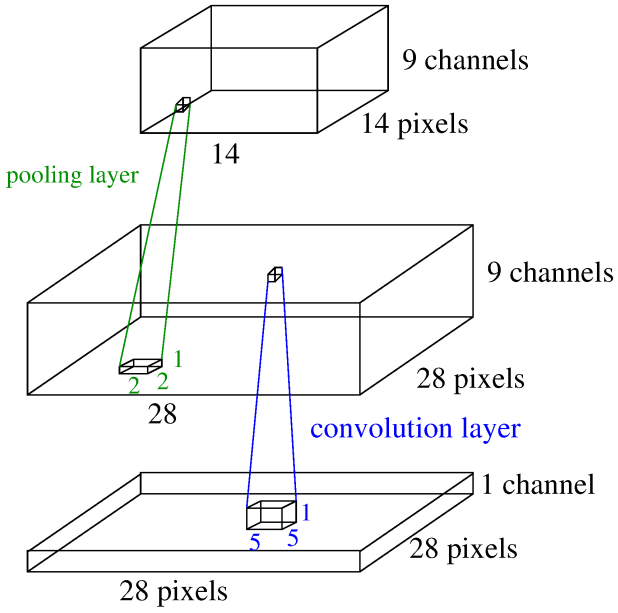
Hence Convolutional

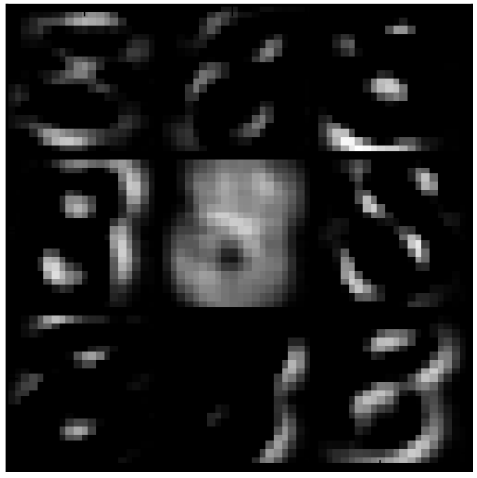
$$h_{x,y,c_1} = \text{relu}\left(\sum_{c_0,\Delta x,\Delta y} W_{\Delta x,\Delta y,c_0,c_1} x_{x+\Delta x,y+\Delta y,c_0} + b_{c_1}\right)$$

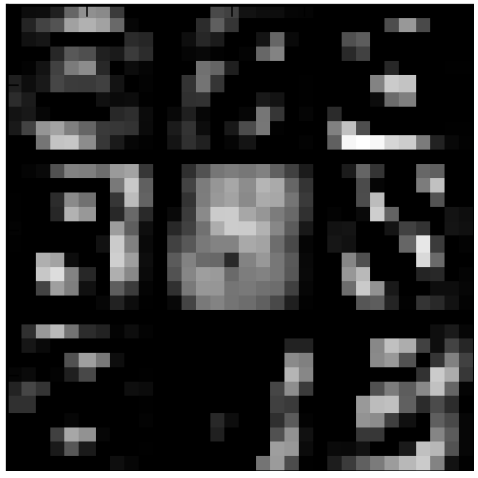
$$\Leftrightarrow \mathbf{h}_{:,:,c_1} = \text{relu}\left(\sum_{c_0} \mathbf{W}_{:,:,c_0,c_1} * \mathbf{x}_{:,:,c_0} + b_{c_1}\right)$$

- Can be written using the convolution operator $*$.
- (Convolution involves flipping one of the two inputs around.
- This is not needed, so the actual operation is cross-correlation.)
- **Padding:** Input can be padded with zeroes to keep the same size.

Pooling

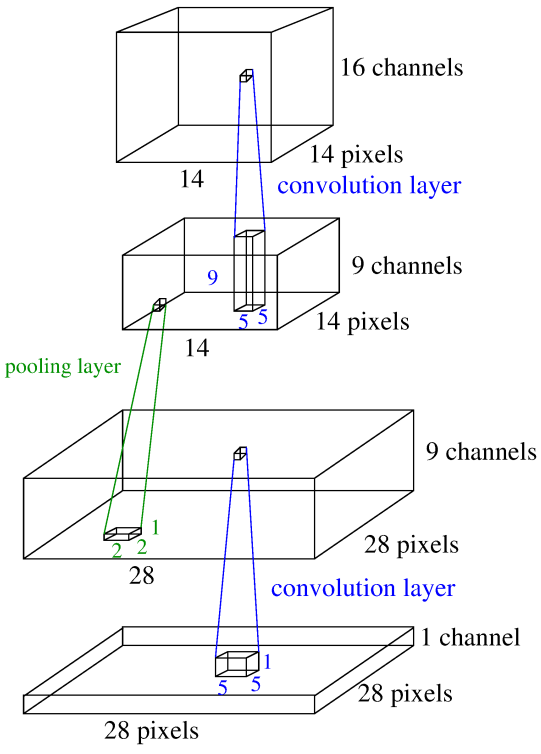


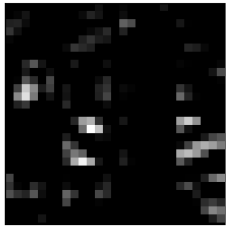


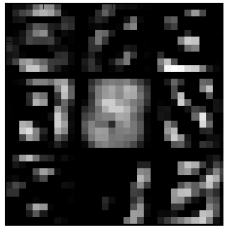


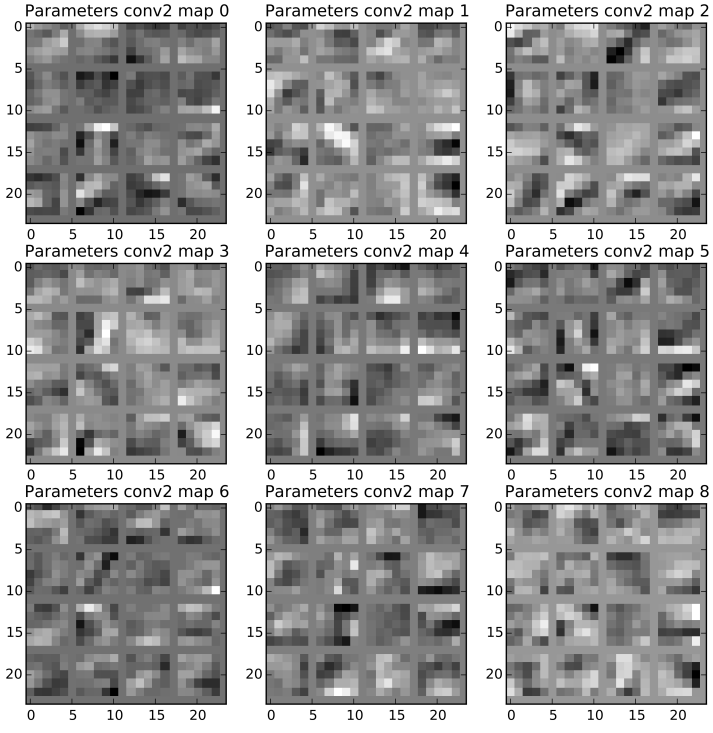
- Decrease resolution and increase channels going up.
- Often down-sampling by hard-coded pooling layers.
- We use $\max(\cdot)$ of 2×2 activations.
- Analysis of pooling functions (Boureau et al., 2010)

Stack more layers



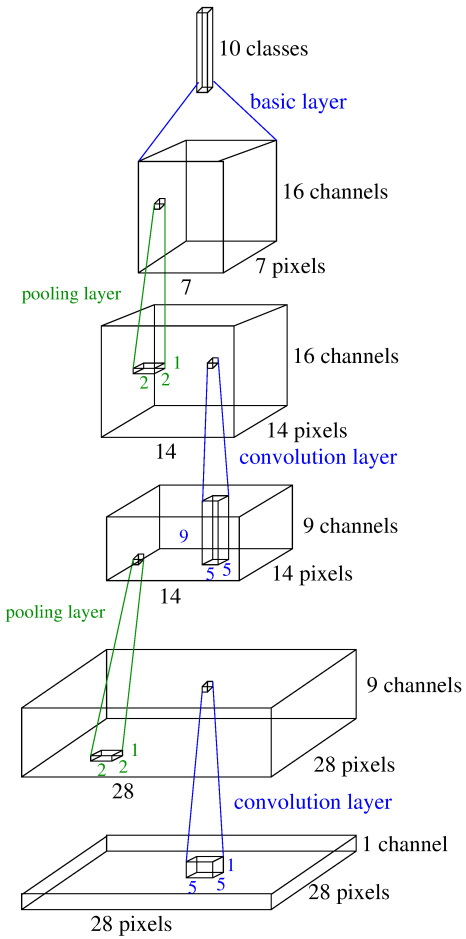


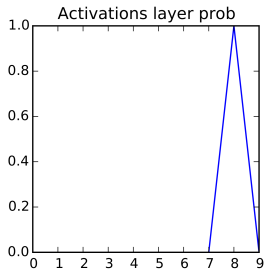


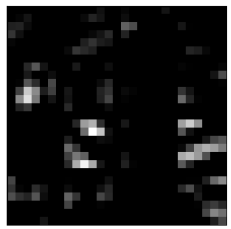


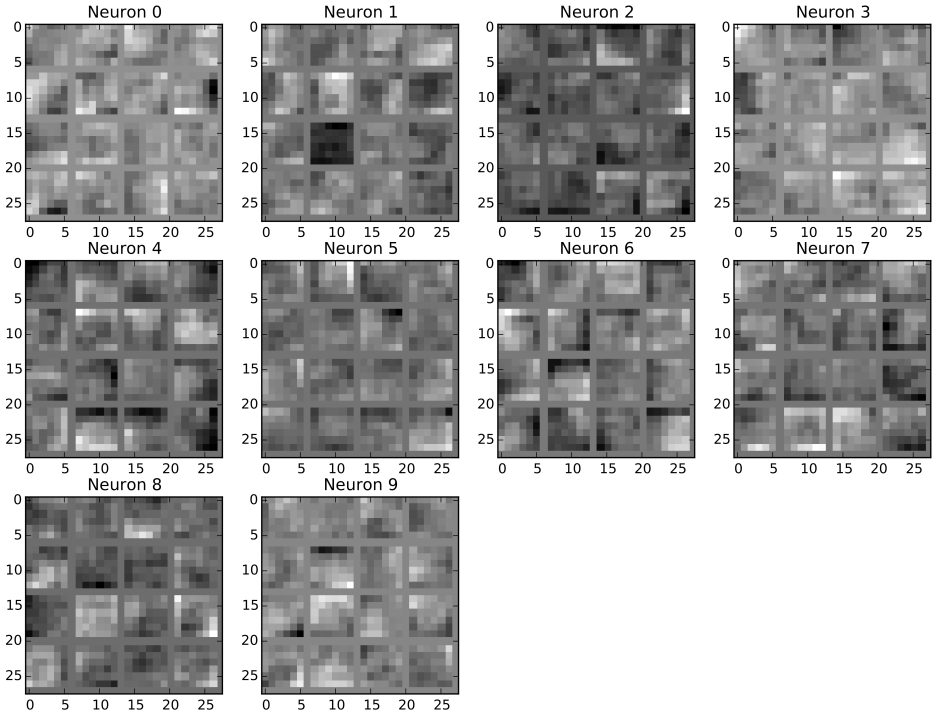
- Note how each unit looks at all channels of the previous layer.

Whole network



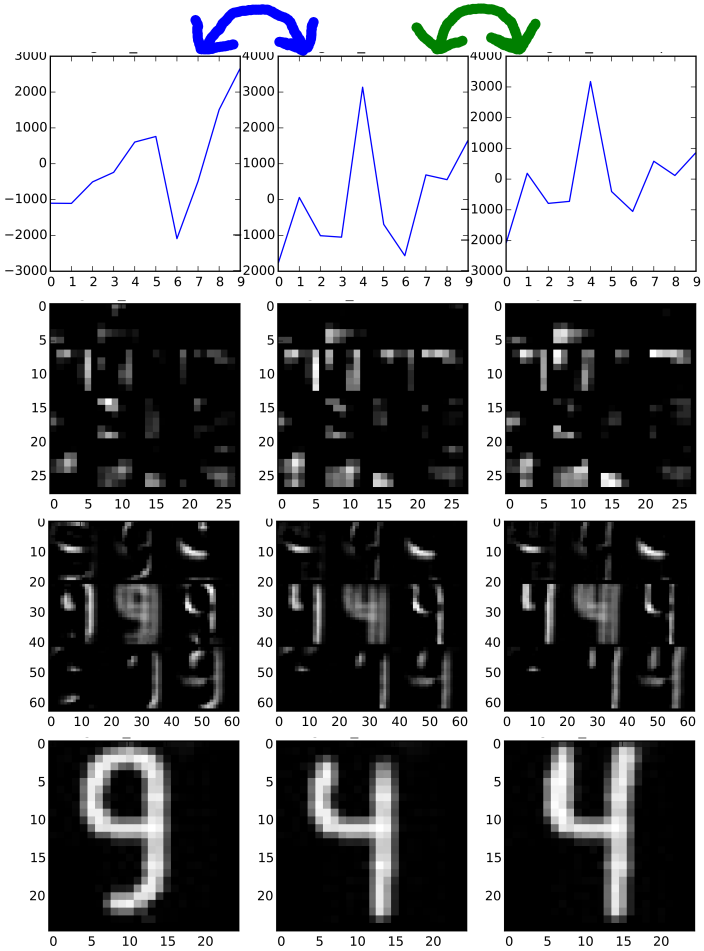


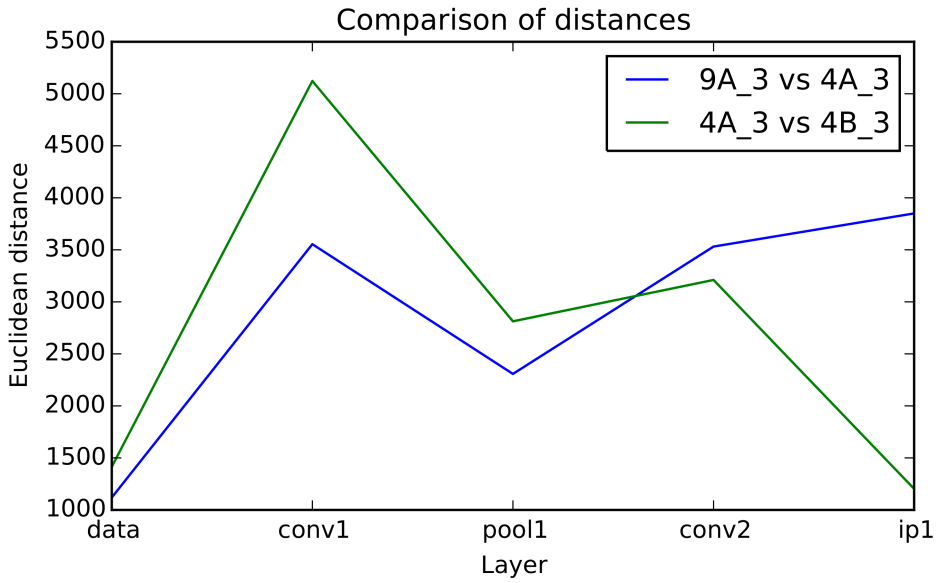




- At the top, the resolution drops to
- 1×1 pixel.
- Train by back-propagation as before.

Good representation?





- In pixel space, 9–4 is closer.
- In feature space, 4–4 is closer.

Convolutional vs. non-convolutional

- Number of weights (ignoring biases):

  $5 \cdot 5 \cdot 9 + 5 \cdot 5 \cdot 9 \cdot 16 + 7 \cdot 7 \cdot 16 \cdot 10 = 225 + 3600 + 7840 = 11665$
- Sizes of signals $\mathbf{h}$:

  $28 \cdot 28, 28 \cdot 28 \cdot 9, 14 \cdot 14 \cdot 16 = 784, 7056, 3136$

Convolutional vs. non-convolutional

- Number of weights (ignoring biases):

  $5 \cdot 5 \cdot 9 + 5 \cdot 5 \cdot 9 \cdot 16 + 7 \cdot 7 \cdot 16 \cdot 10 = 225 + 3600 + 7840 = 11665$
- Sizes of signals $\mathbf{h}$:

  $28 \cdot 28, 28 \cdot 28 \cdot 9, 14 \cdot 14 \cdot 16 = 784, 7056, 3136$

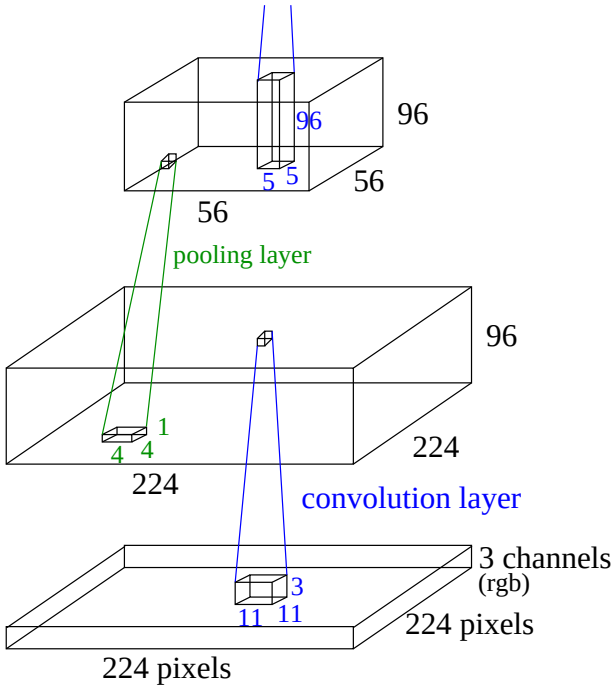
- Compare to the introductory MNIST FFNN example
- Weights:

  $28 \cdot 28 \cdot 225 + 225 \cdot 144 + 144 \cdot 10 =$
  $176400 + 32400 + 1440 = 210240$
- Signals:

  $784, 225, 144$

- Convolutional network has more signals but less params.

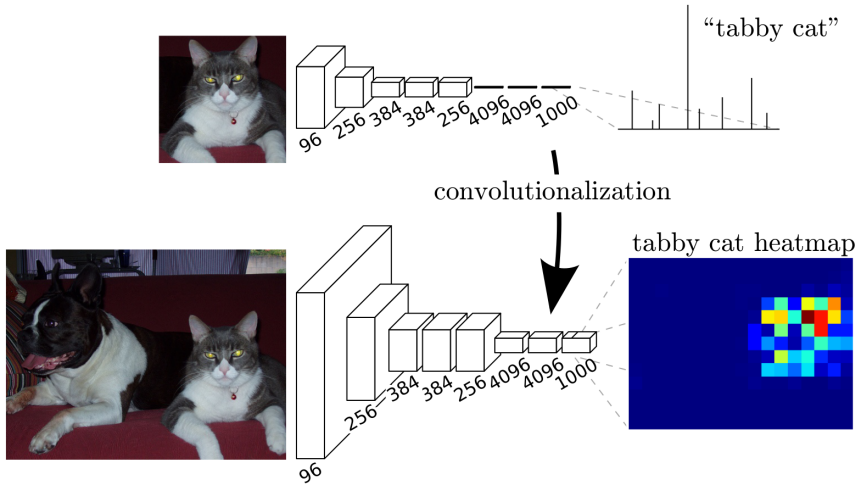
Scaling up

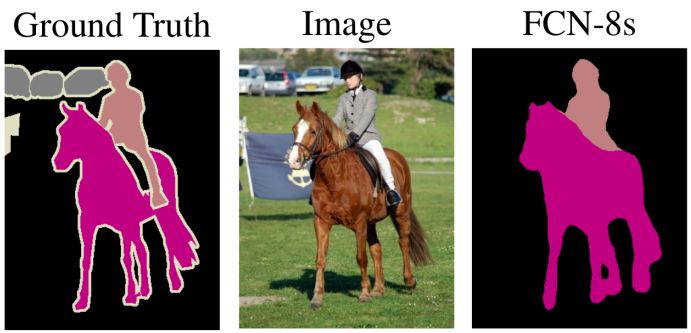


Ipython notebook example

- First layers of (Krizhevsky, 2012).
- Trend in 2015 towards small filters (3×3) and poolings (2×2),
- but lots of layers (10–40).

Semantic segmentation (Long et al., 2015)

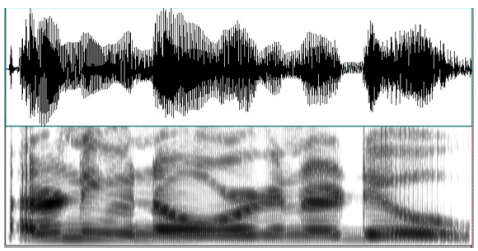


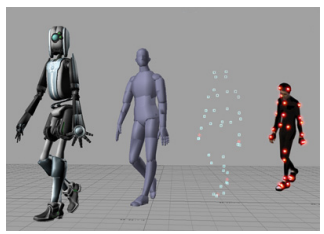


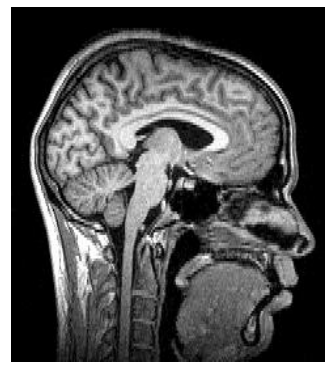
- Sliding a convolutional network to classify each location.
- Lots of shared computation.

Other applications

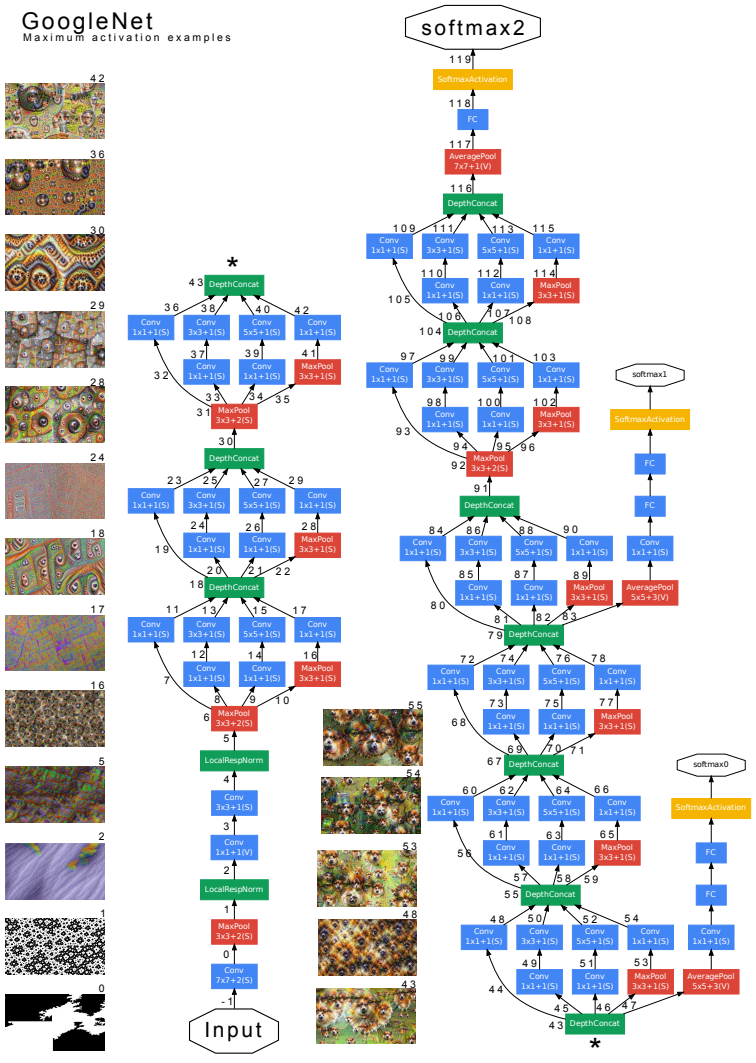
Tensor	Single channel	Multi-channel
1-D	Raw audio (mono)	Motion capture
2-D	Audio + Fourier transform	Game of Go
3-D	Brain imaging	Colour video

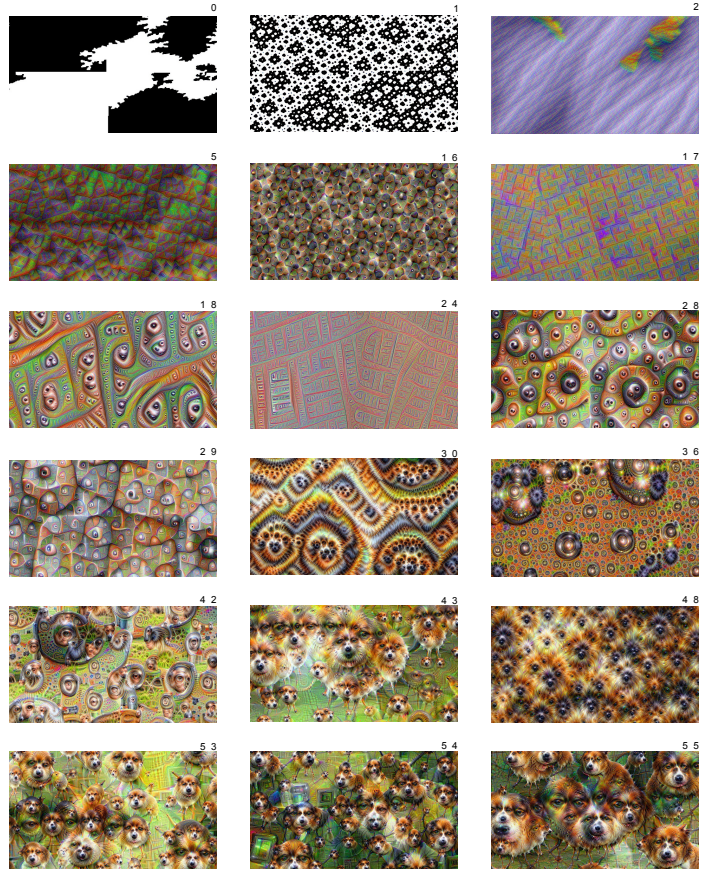






Inceptionism (Mordvintsev, 2015) Video by Miquel Perello Nieto





https://youtube.com/watch?v=w5U7EL72ngI

users.ics.aalto.fi/perellm1/deep_dreams.shtml

References regularisation

- Tikhonov, Andrey Nikolayevich (1943). On the stability of inverse problems (in Russian). Doklady Akademii Nauk SSSR 39 (5): 195–198.
- J. Luketina, M. Berglund, and T. Raiko (2016). Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters. To be presented at the ICLR workshop track. Preprint available as arXiv:1511.06727 [cs.LG].
- Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., & Brox, T. (2014). Discriminative unsupervised feature learning with convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 766-774).
- Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. Neural networks, 4(1), 67–79.
- Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In ICML, pp. 833-840, 2011.
- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. ”Distilling the knowledge in a neural network.” NIPS 2014 Deep Learning Workshop.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus. Intriguing properties of neural networks. International Conference on Learning Representations (2014)
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii. Distributional Smoothing with Virtual Adversarial Training. Under review as a conference paper at ICLR 2016.

References CNNs

- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.
- Boureau, Y-Lan, Jean Ponce, and Yann LeCun. "A theoretical analysis of feature pooling in visual recognition." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." arXiv preprint arXiv:1411.4038 (2015).
- Mathieu, Michael, Mikael Henaff, and Yann LeCun. "Fast training of convolutional networks through FFTs." arXiv preprint arXiv:1312.5851 (2013).
- Kivinen, Jyri J., and Christopher KI Williams (2011). "Transformation equivariant Boltzmann machines." Artificial Neural Networks and Machine Learning (ICANN).
- Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." Computer Vision–ECCV 2014. Springer International Publishing, 2014. 818-833.
- Mordvintsev, A., Olah, C., & Tyka, M. (2015) "Inceptionism: Going Deeper into Neural Networks", Google research blogspot

Thanks!
Ole Winther