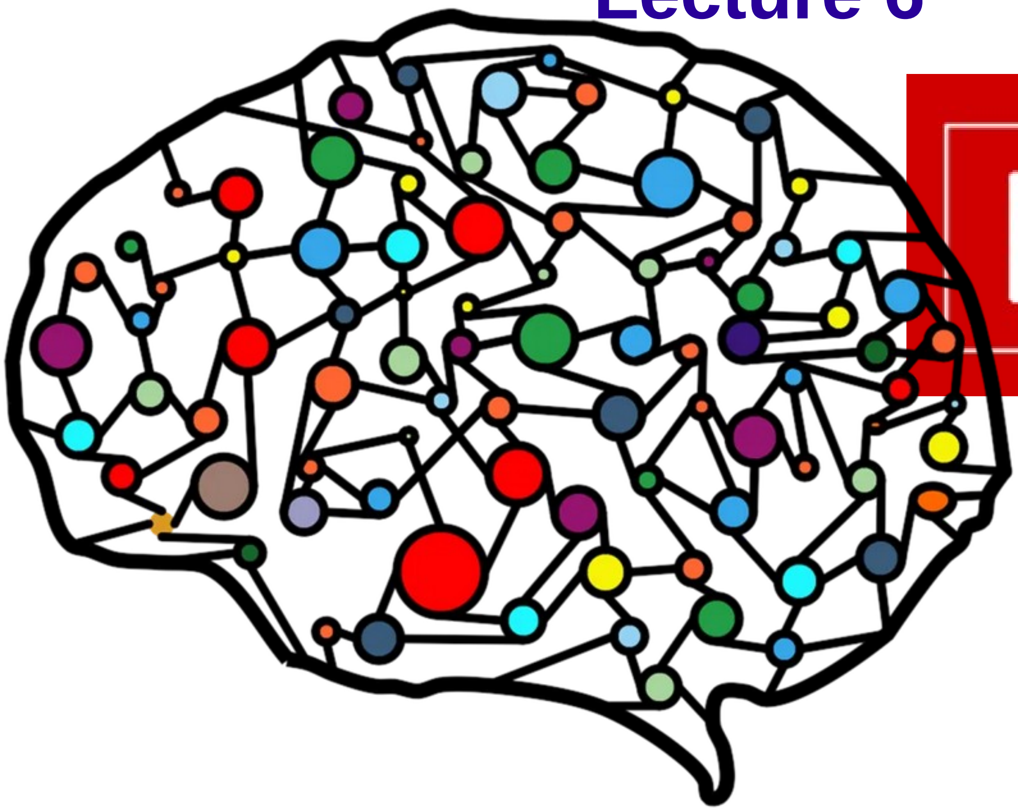
Machine learning
Lecture 6



- Convolutional Deep Neural Network

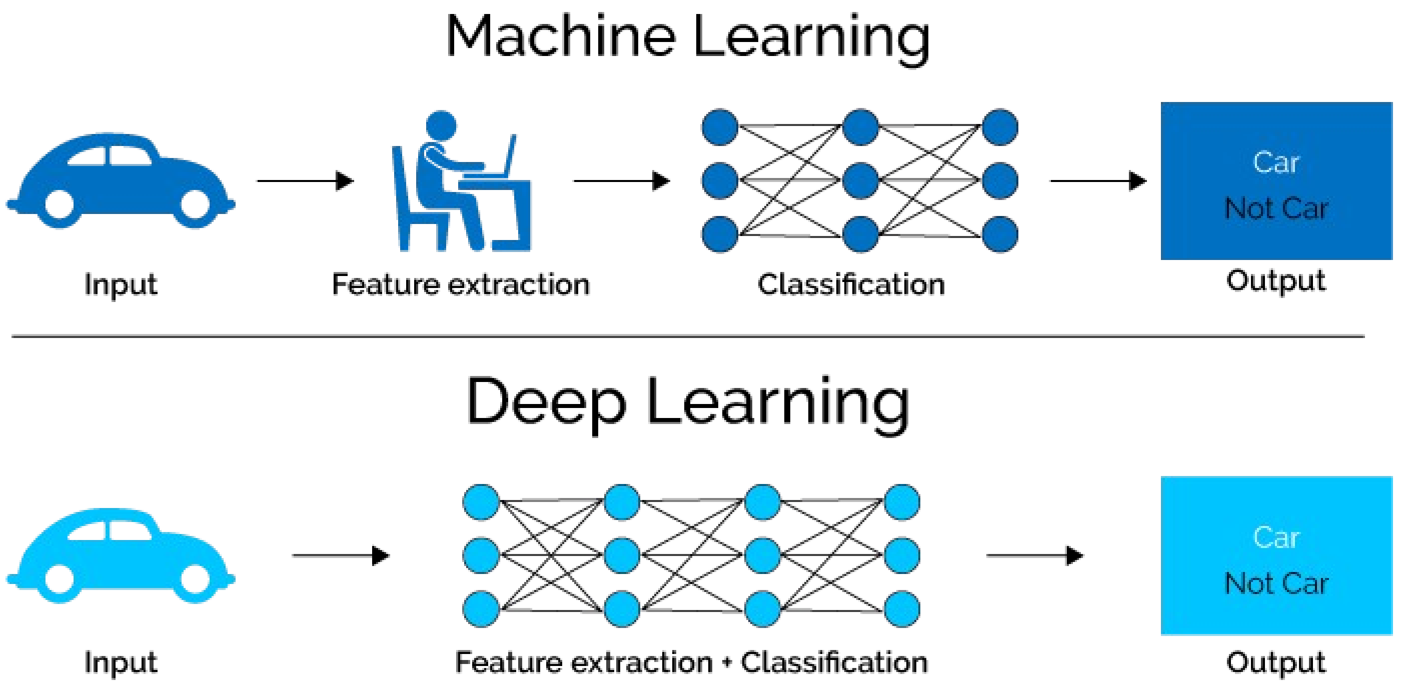
Marcin Wolter

IFJ PAN

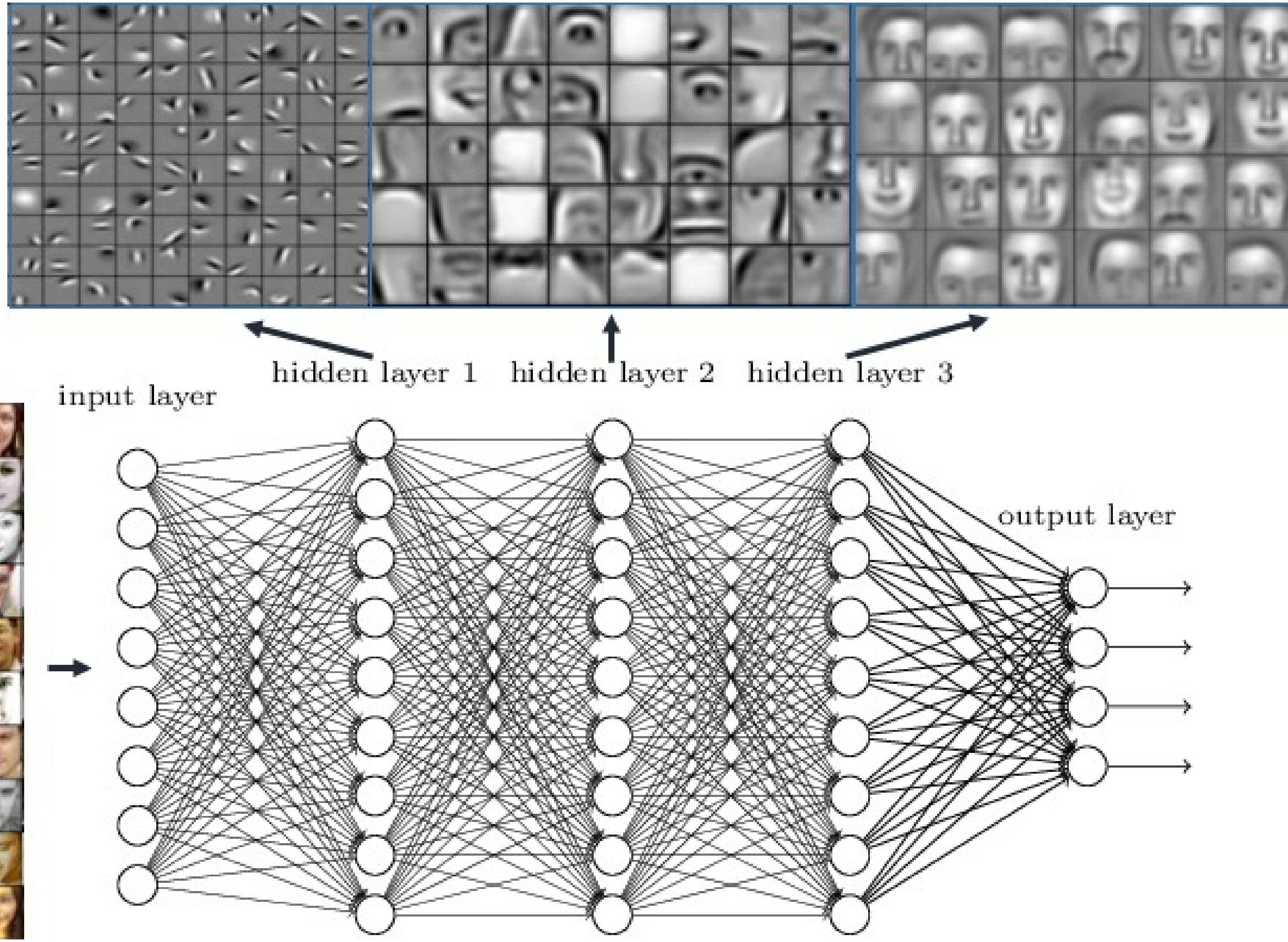
2 December 2020

Machine Learning and Deep Learning

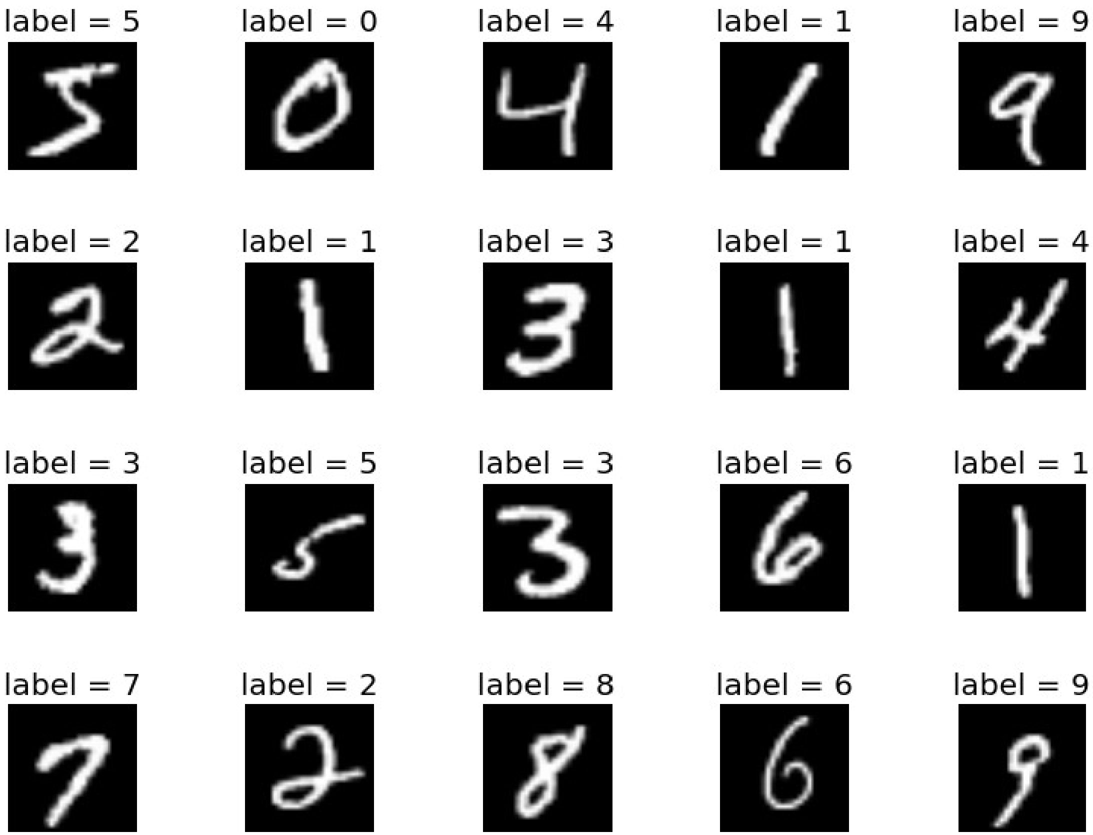
- **Traditional ML (BDT, NN etc)** – the scientist finds good, well discriminating variables (~10), called “features”, and performs classification using them as inputs for the ML algorithm.
- **Deep Learning** – thousands or millions of input variables (like pixels of a photo), the features are *automagically* extracted during training.



Deep neural networks learn hierarchical feature representations



We have designed a net to identify hand-written digits



28 x 28 pixels

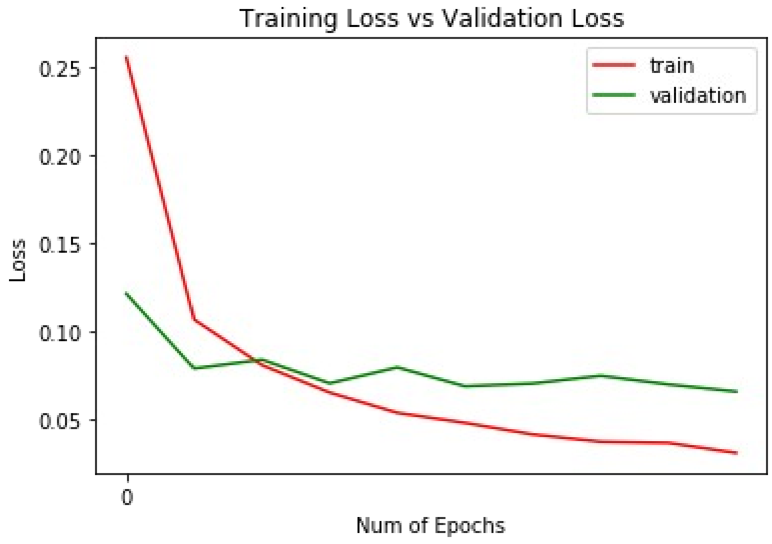
```
60000 train samples
10000 test samples
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                5130
=================================================================
Total params: 932,362
Trainable params: 932,362
Non-trainable params: 0
```

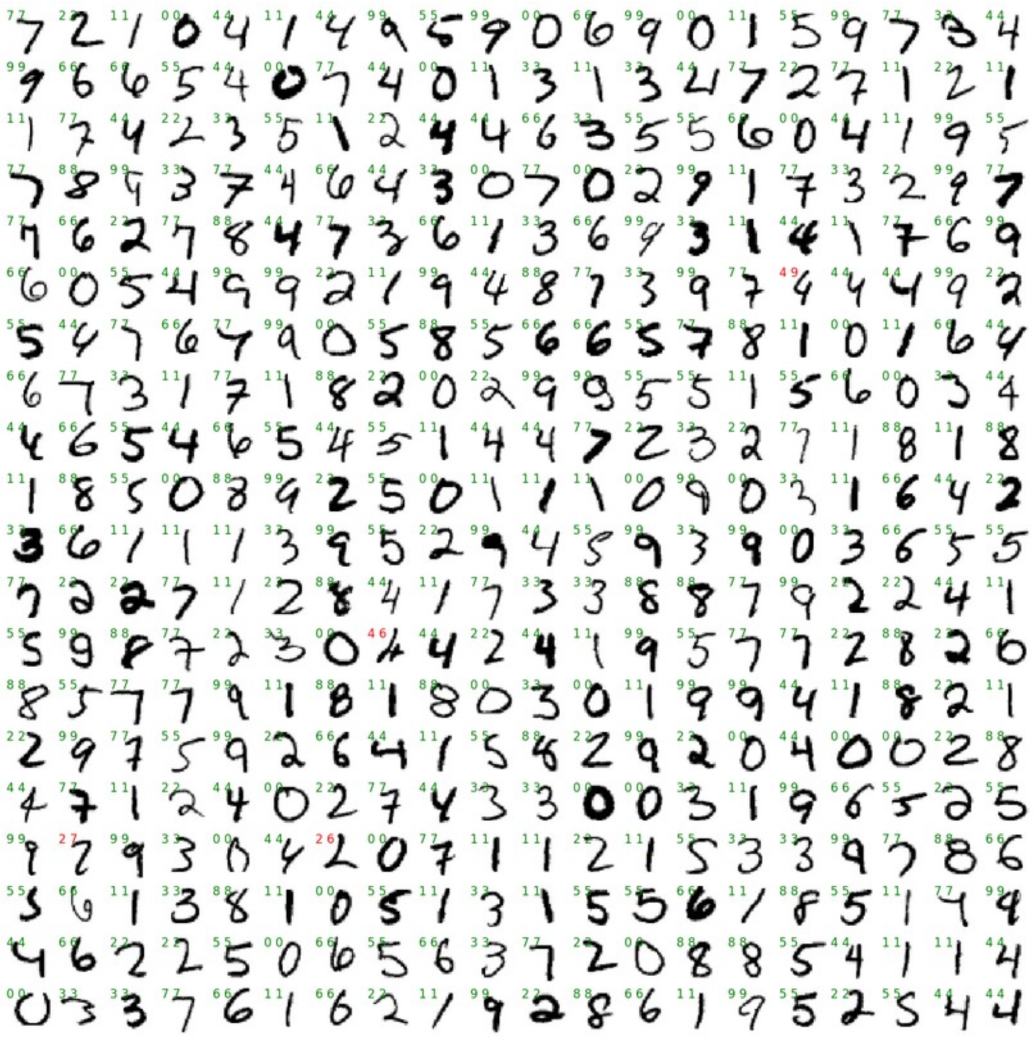
Program with all features

- https://github.com/marcinwolter/DeepLearning_2020/blob/main/mnist_mlp.ipynb

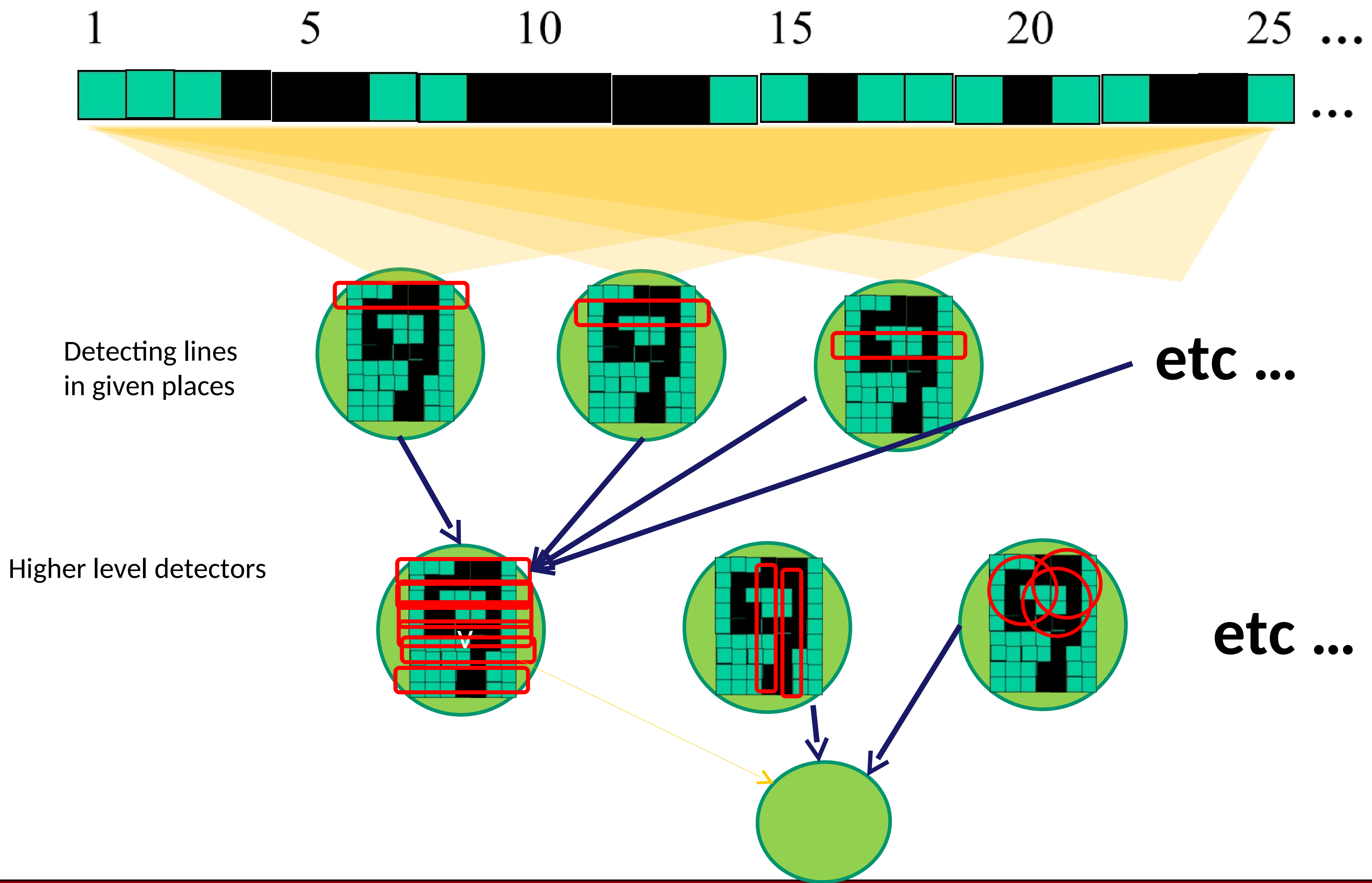
- Visualization of results

- Plotting the Neural Network structure



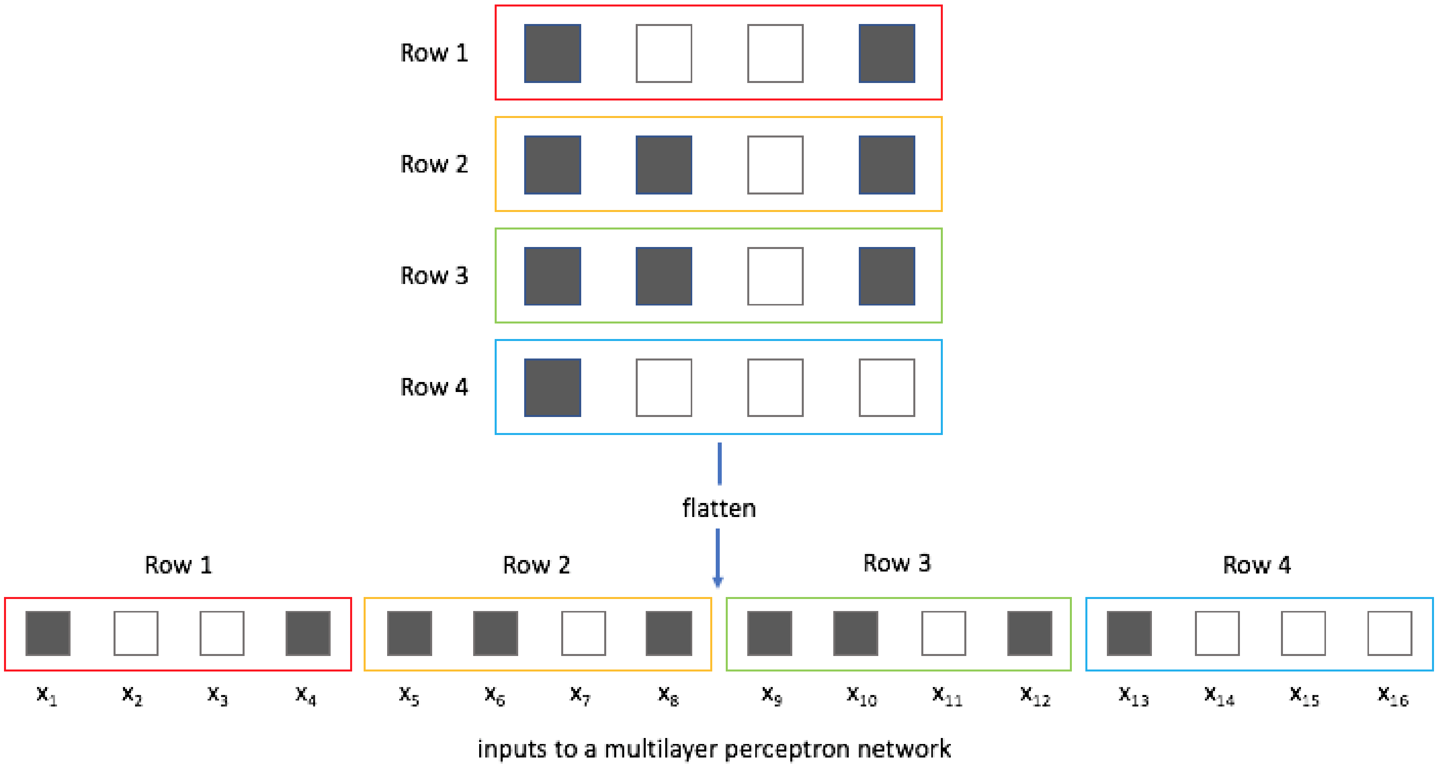


Each layer is finding more and more complicated features



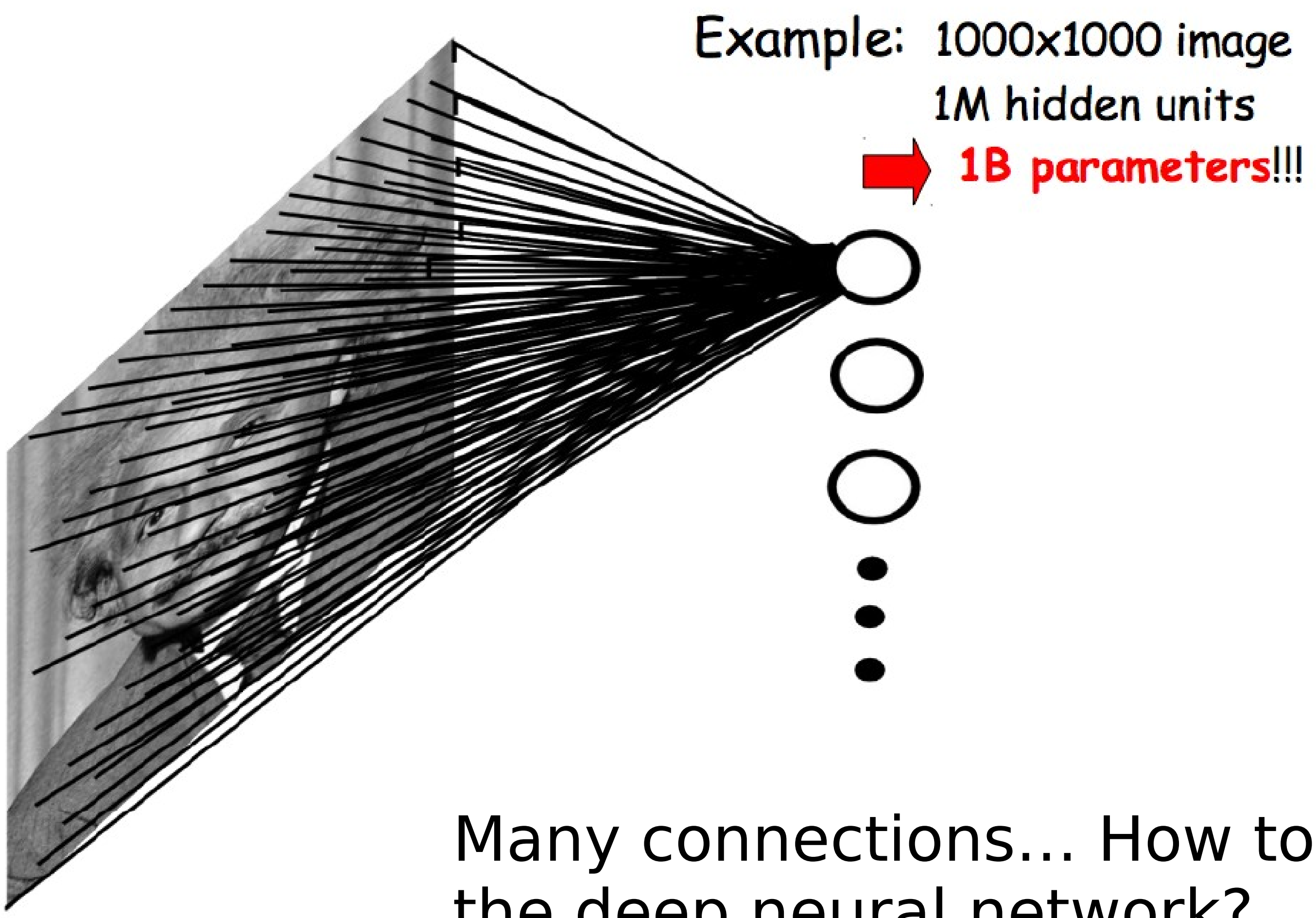
Picture to vector

- A picture (array of pixels) is transformed to an input vector (flattening):



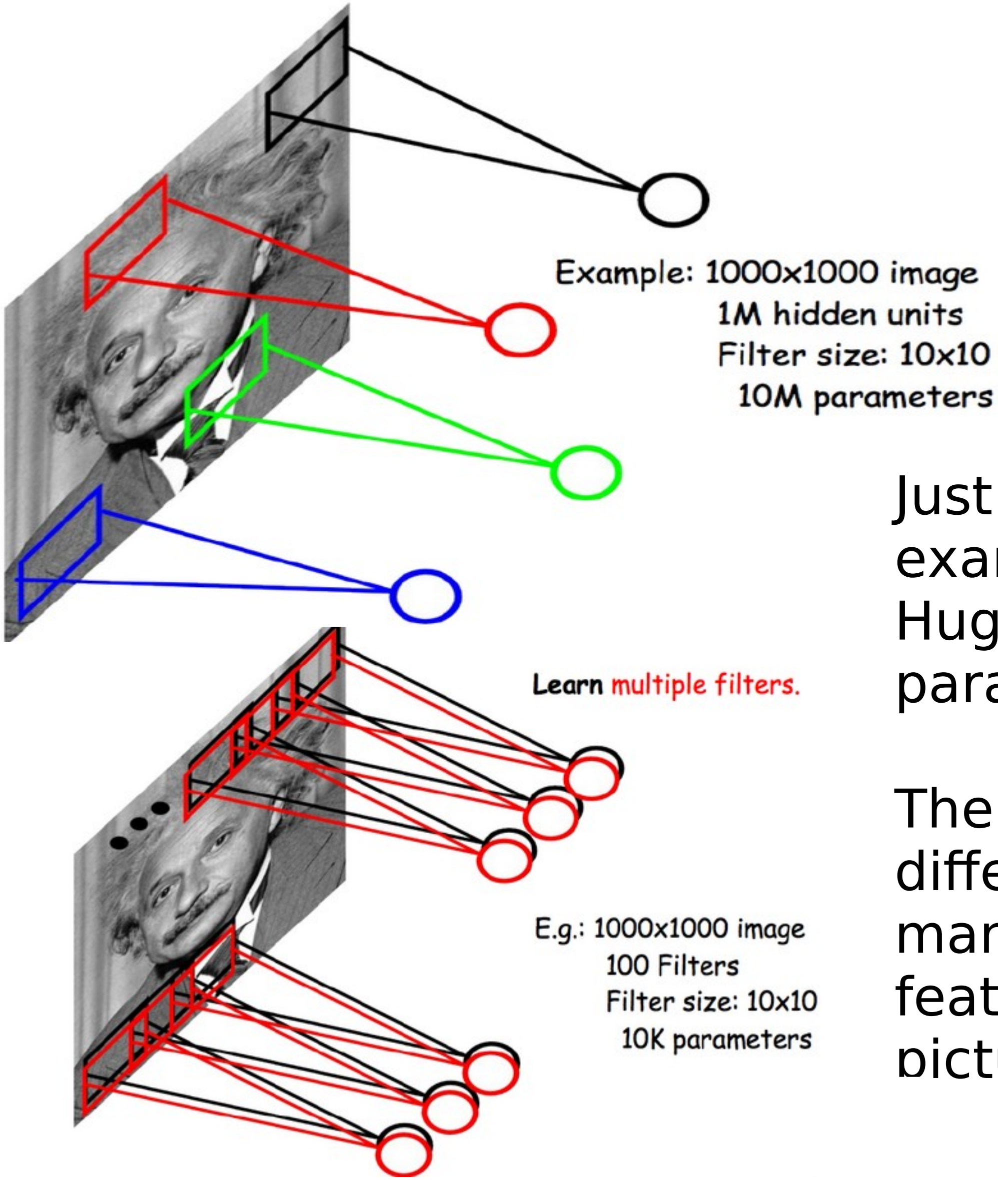
Loss of information – neighbour pixels are no longer close together.

Convolutional NN
Pattern recognition



Many connections... How to simplify the deep neural network?

Convolutional NN

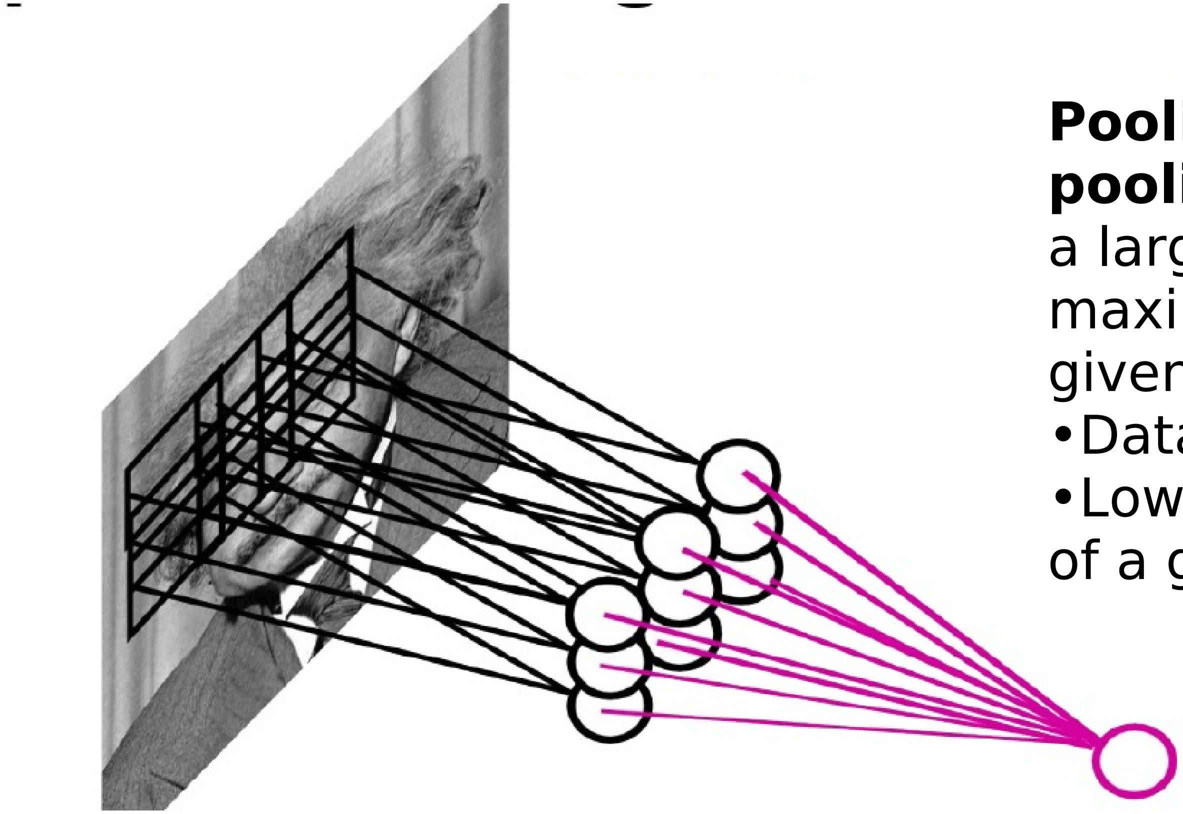


Just connect only local areas, for example 10x10 pixels.
Huge reduction of the number of parameters!

The same features might be found in different places => so we could train many filters, each recognizing another feature, and move them over the picture.

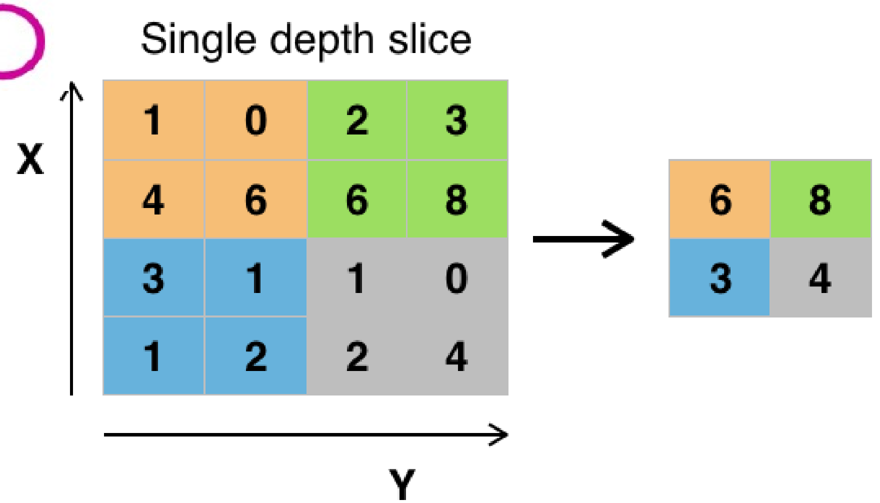
LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

Pooling



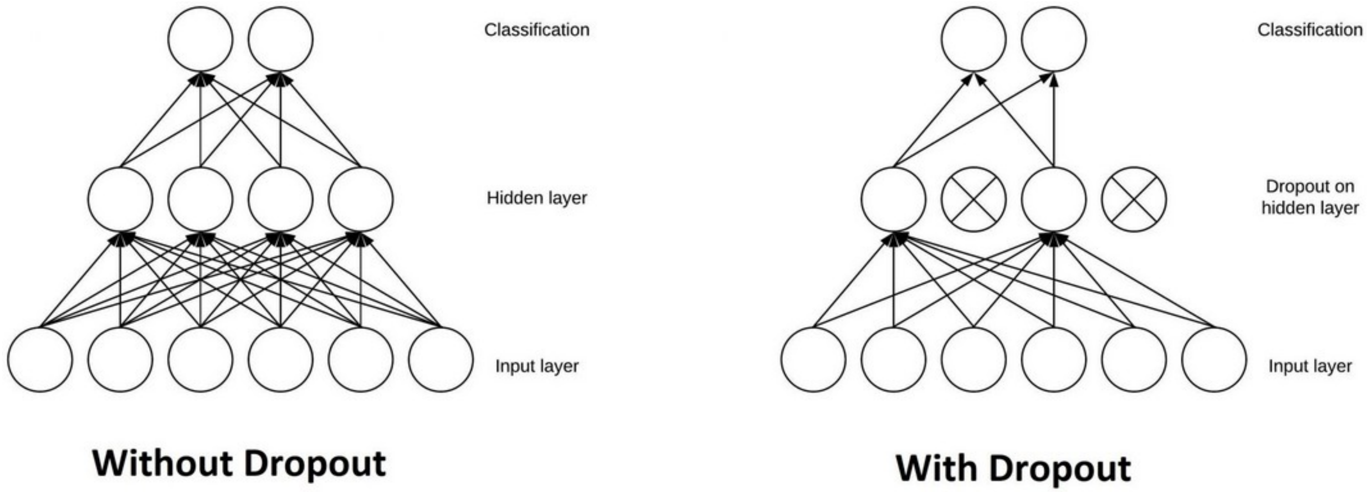
Pooling – (in most cases **max pooling**) the group of outputs for a larger input area is replaced by a maximum (or average) for this given area:

- Data reduction,
- Lower sensitivity for the position of a given feature.



Dropout

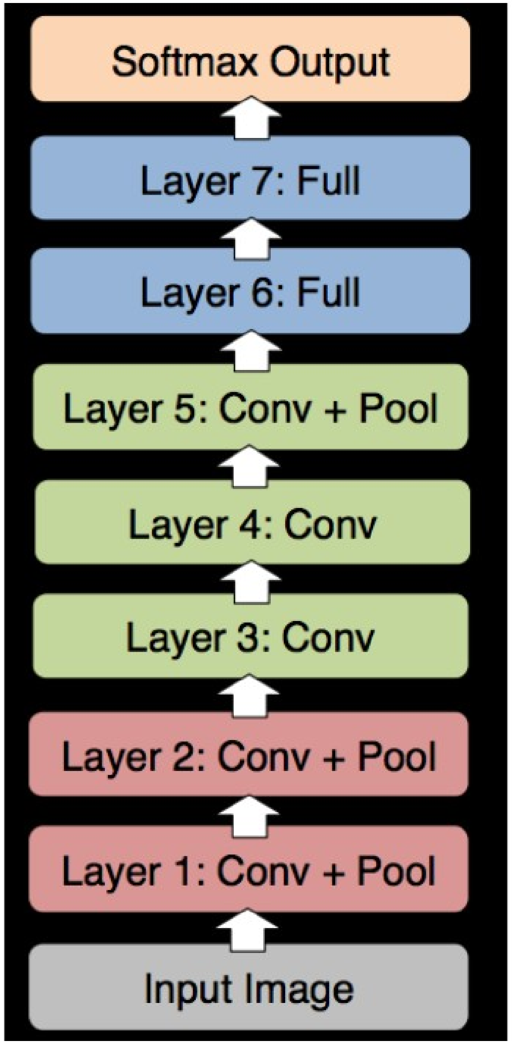
- Effective and most commonly used regularization techniques for neural networks, developed by Hinton and his students at the University of Toronto. Dropout, applied to a layer, consists of randomly "dropping out" (i.e. setting to zero) a number of output features of the layer during training.



- Geoff Hinton has said that he was inspired, among other things, by a fraud prevention mechanism used by banks -- in his own words: *"I went to my bank. The tellers kept changing and I asked one of them why. He said he didn't know but they got moved around a lot. I figured it must be because it would require cooperation between employees to successfully defraud the bank. This made me realize that randomly removing a different subset of neurons on each example would prevent conspiracies and thus reduce overfitting".*

Architecture of Alex Krizhevsky et al.

- 8 layers total.
- Trained on Imagenet Dataset (1000 categories, 1.2M training images, 150k test images)
- 18.2% top-5 error
  - Winner of the ILSVRC-2012 challenge.

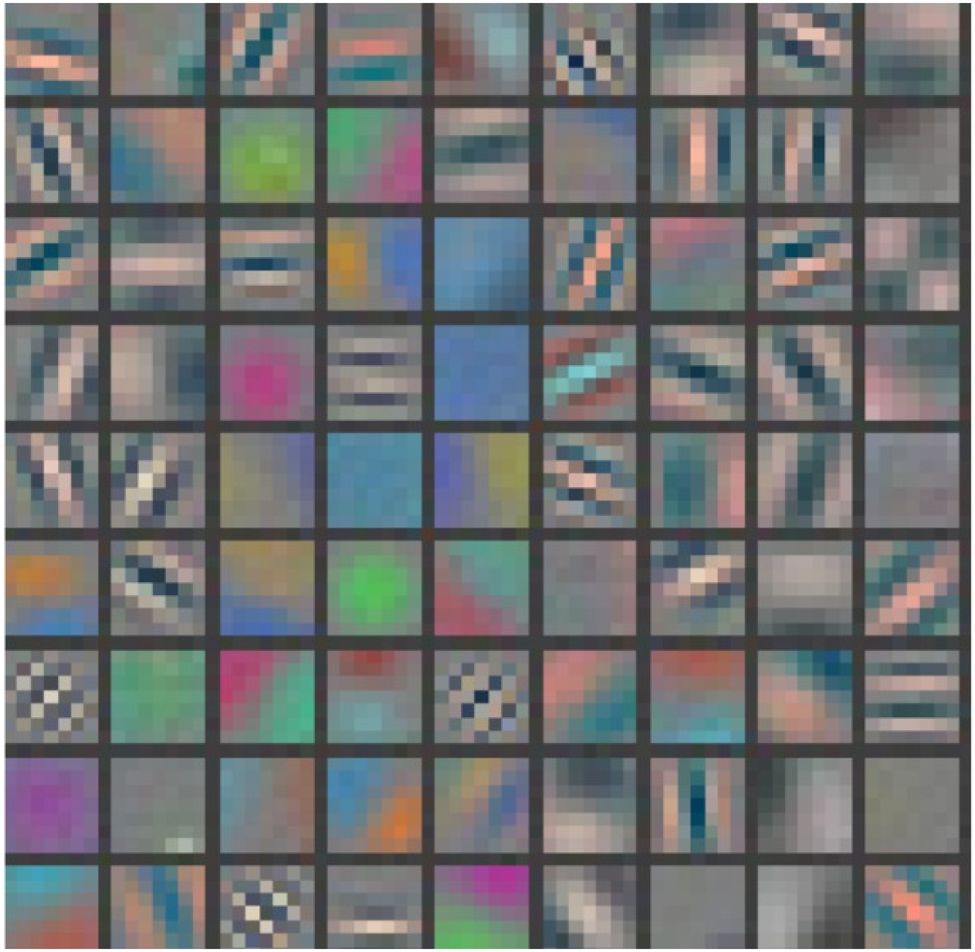


Slide: R. Fergus

First layer filters

Showing 81 filters of 11x11x3.

Capture low-level features like oriented edges, blobs.

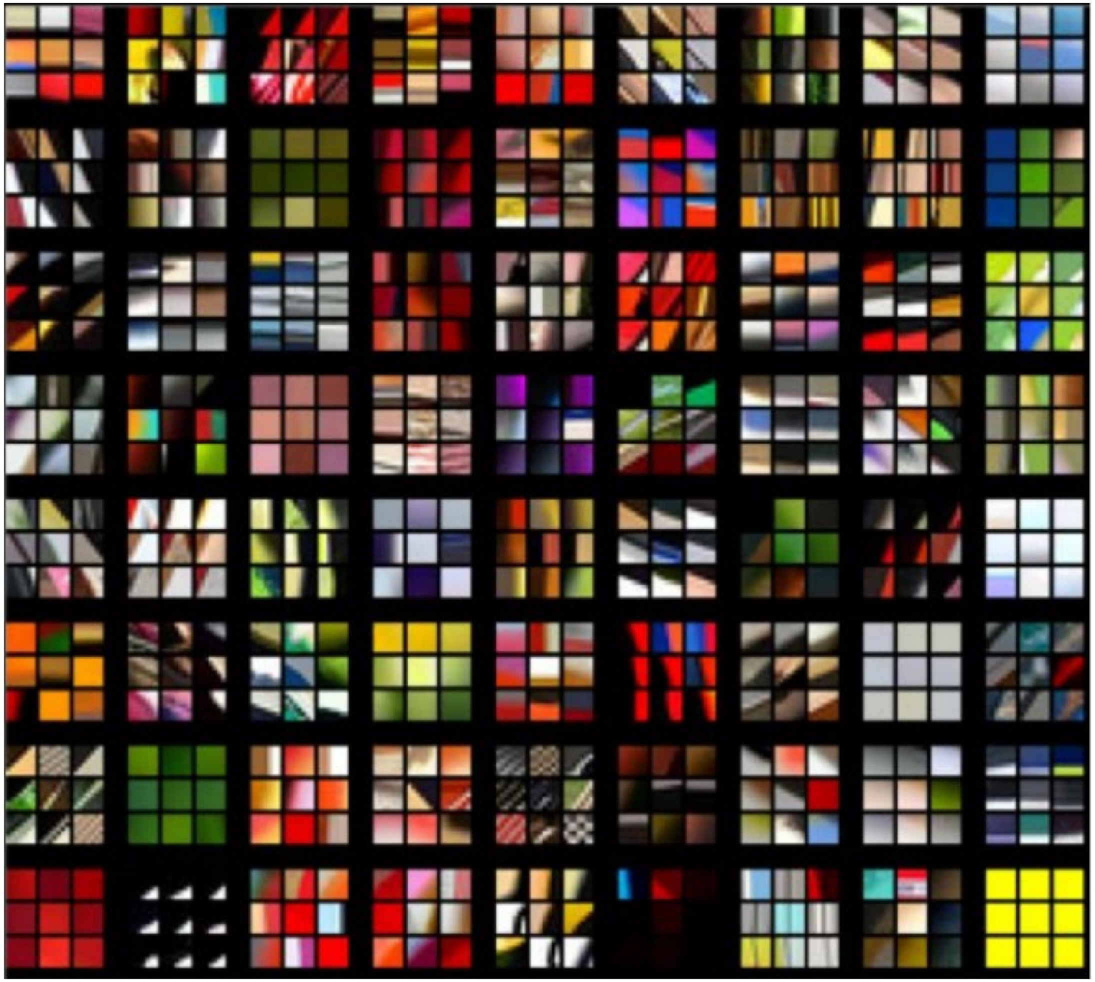


Note these oriented edges are analogous to what SIFT uses to compute the gradients.

SIFT - scale-invariant feature transform, algorithm published in 1999 roku by David Lowe.

Top 9 patches that activate each filter in layer 1

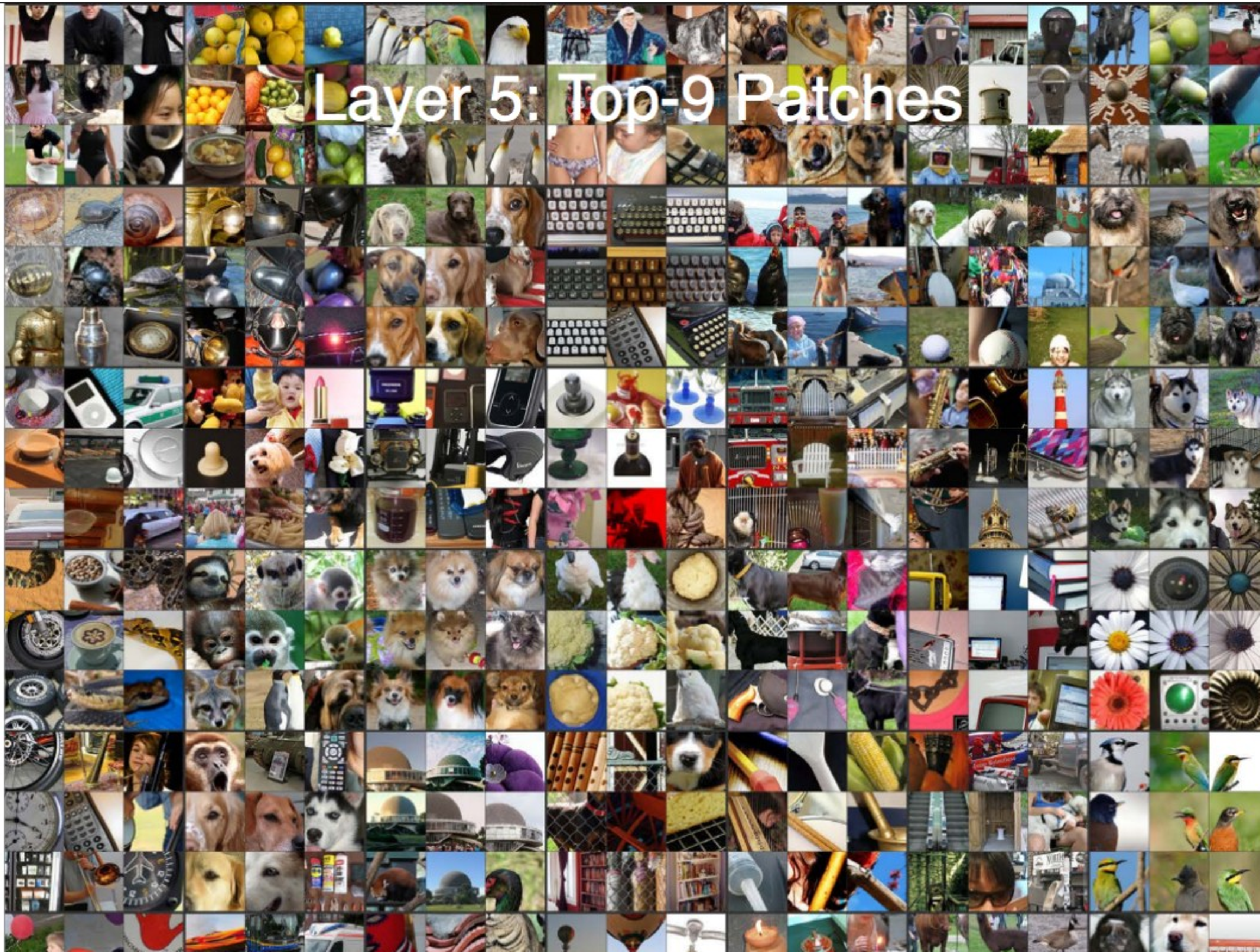
Each 3x3 block shows the top 9 patches for one filter.




Layer 2: Top-9 Patches


Layer 3: Top-9 Patches

Layer 4: Top-9 Patches


Layer 5: Top-9 Patches

Few properties of Deep Neural Networks

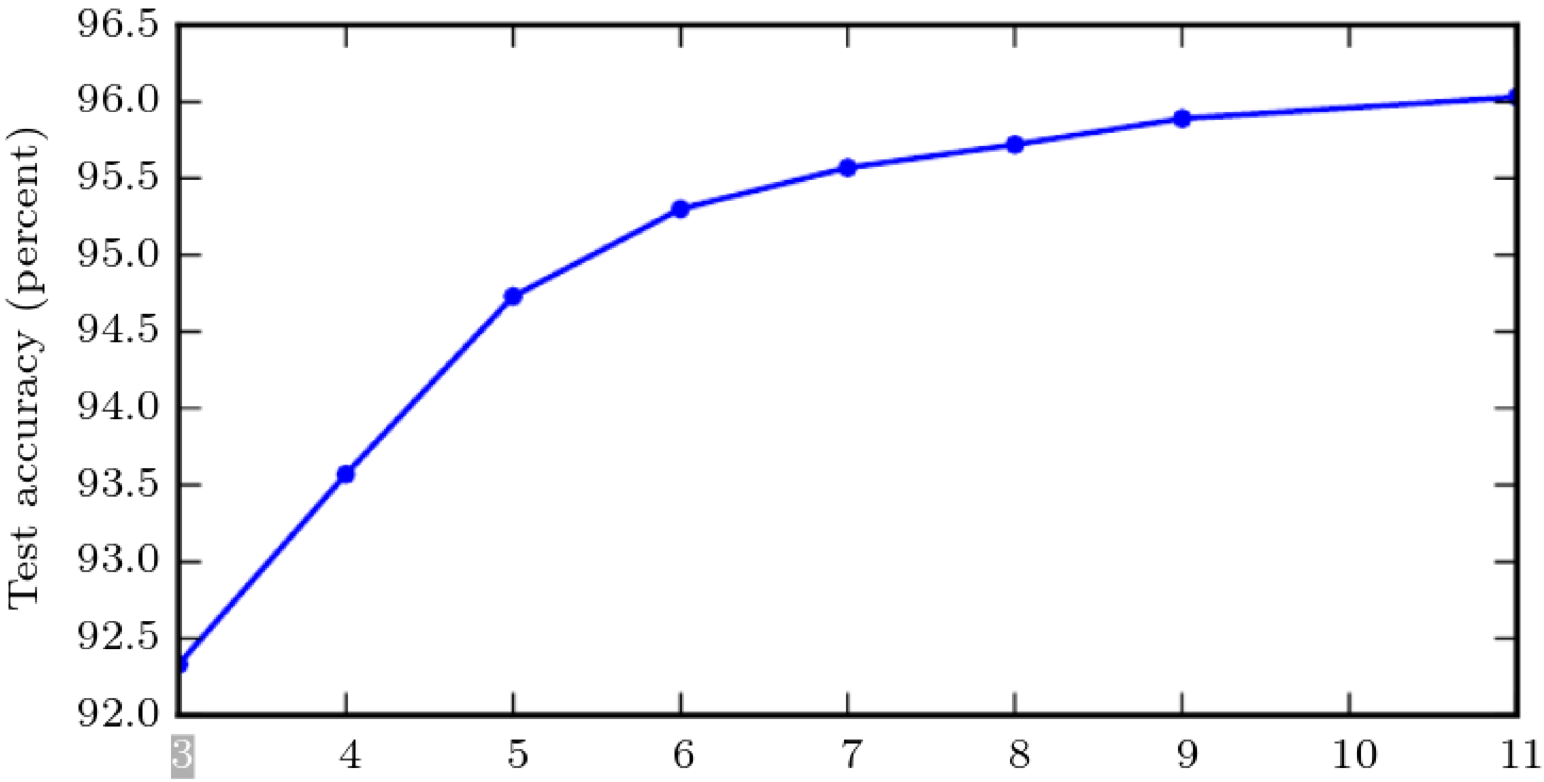


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow et al. (2014d). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

http://www.deeplearningbook.org

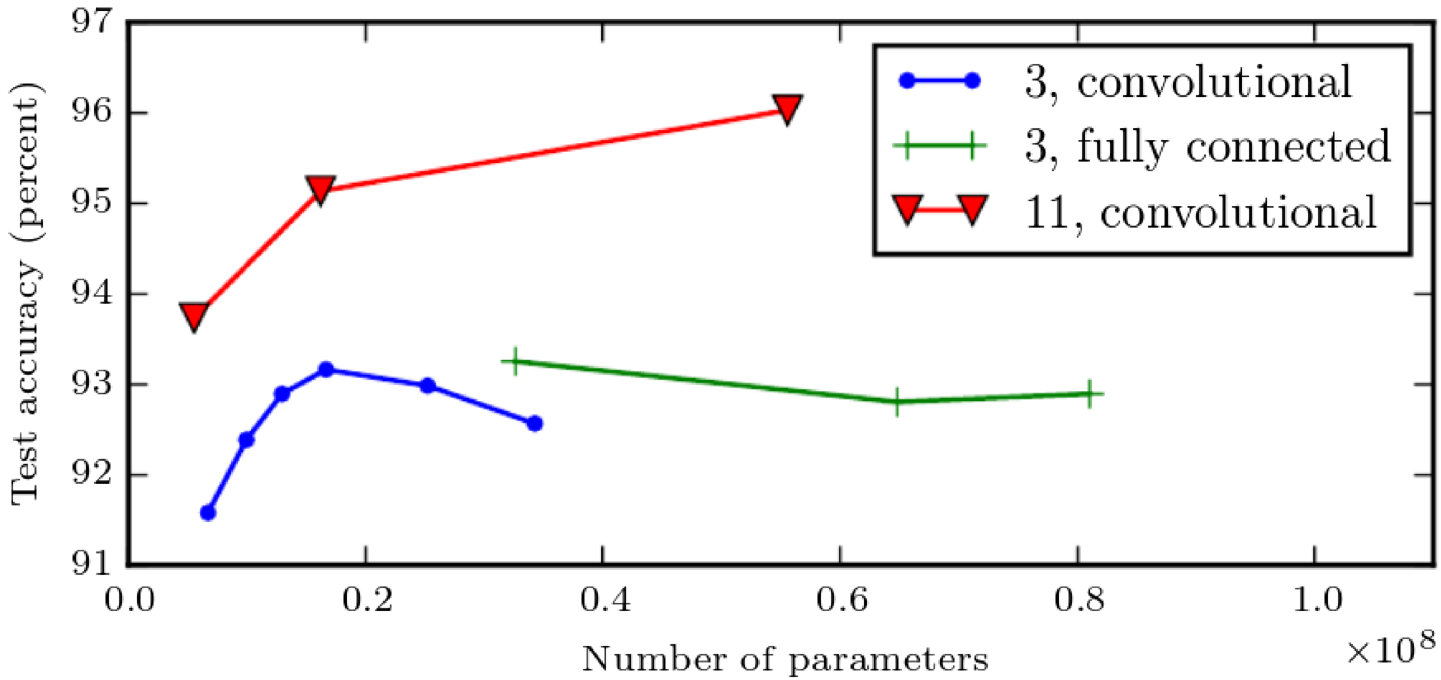


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

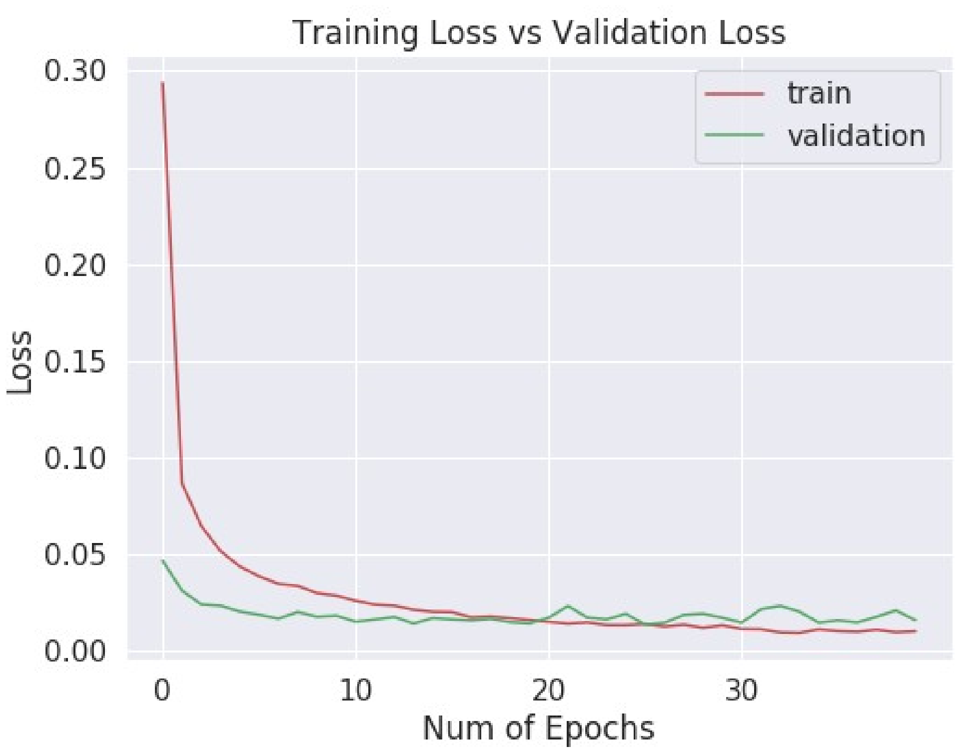
http://www.deeplearningbook.org

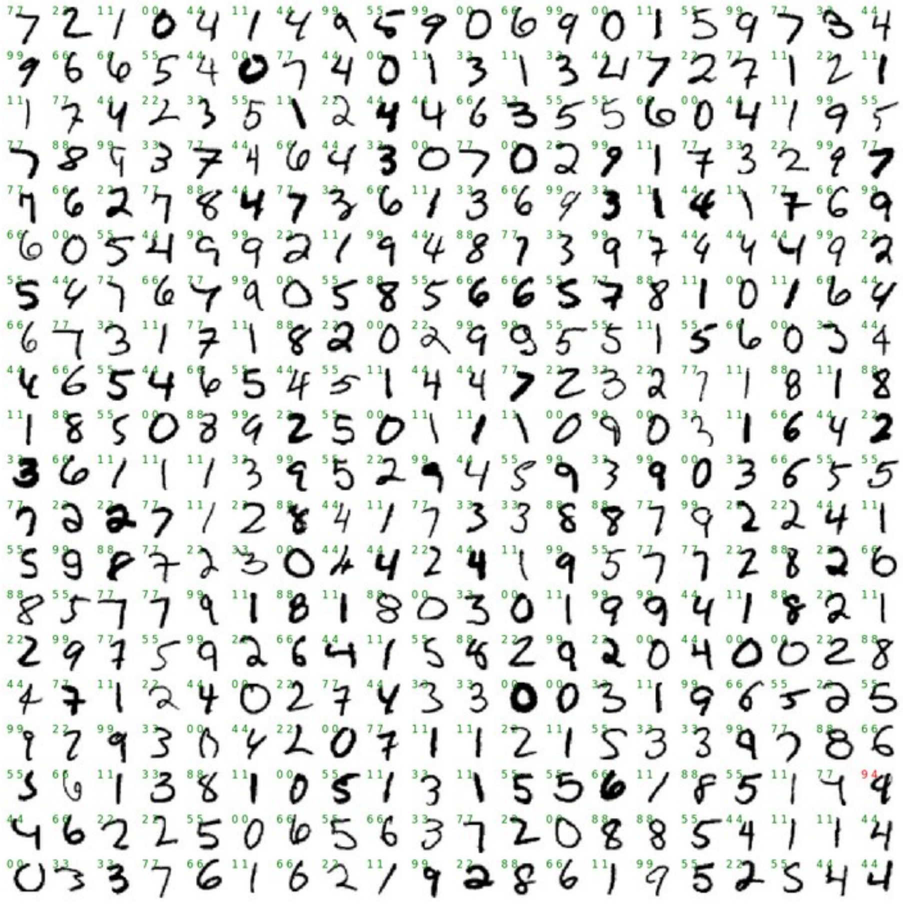
CNN Example

- Back to our digits recognition with CNN…
- Gets much, much better results! Think why?

```
Test loss:     0.0160
Test accuracy: 0.9959
```

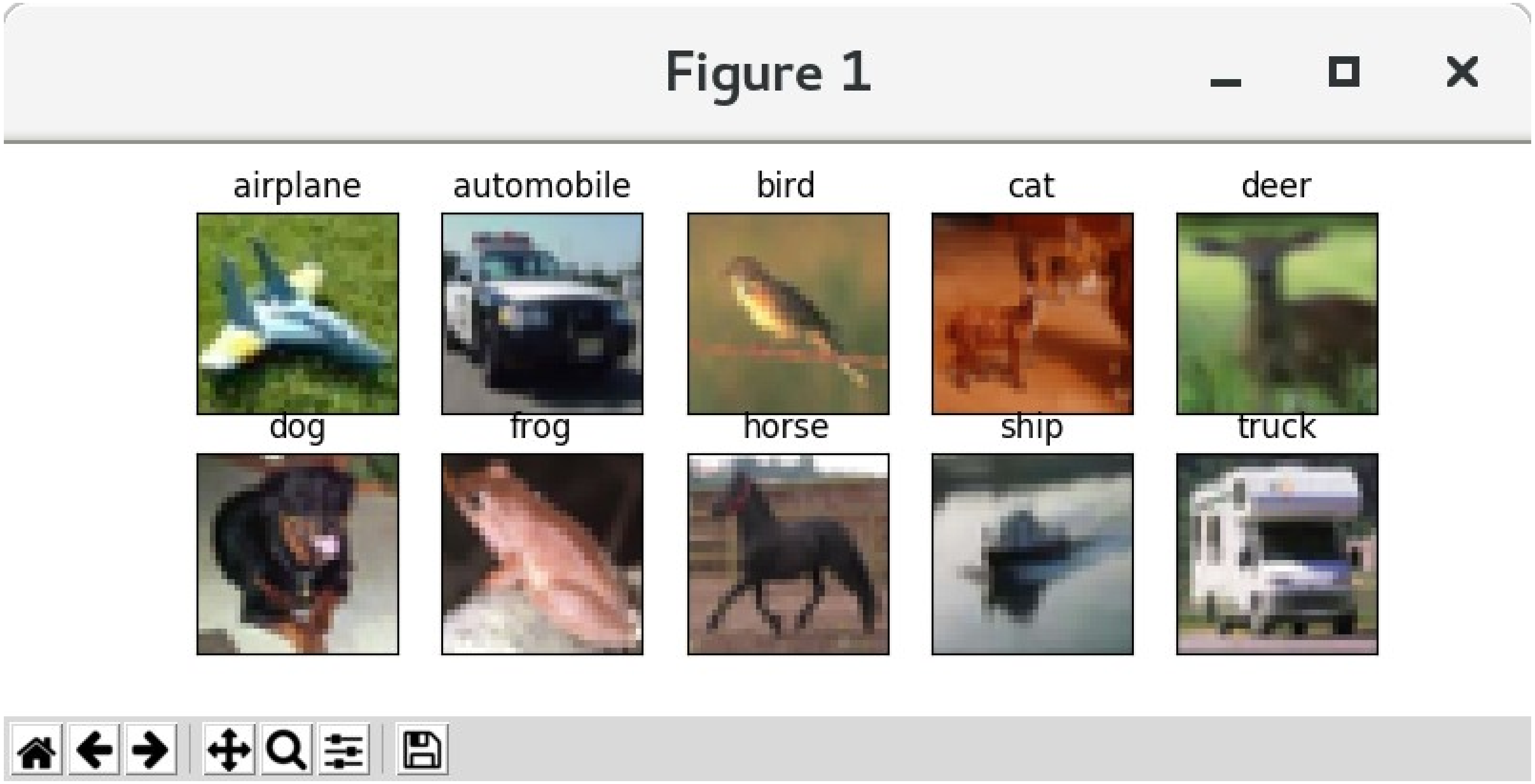
https://github.com/marcinwolter/MachineLearning2020/blob/main/mnist_cnn.ipynb





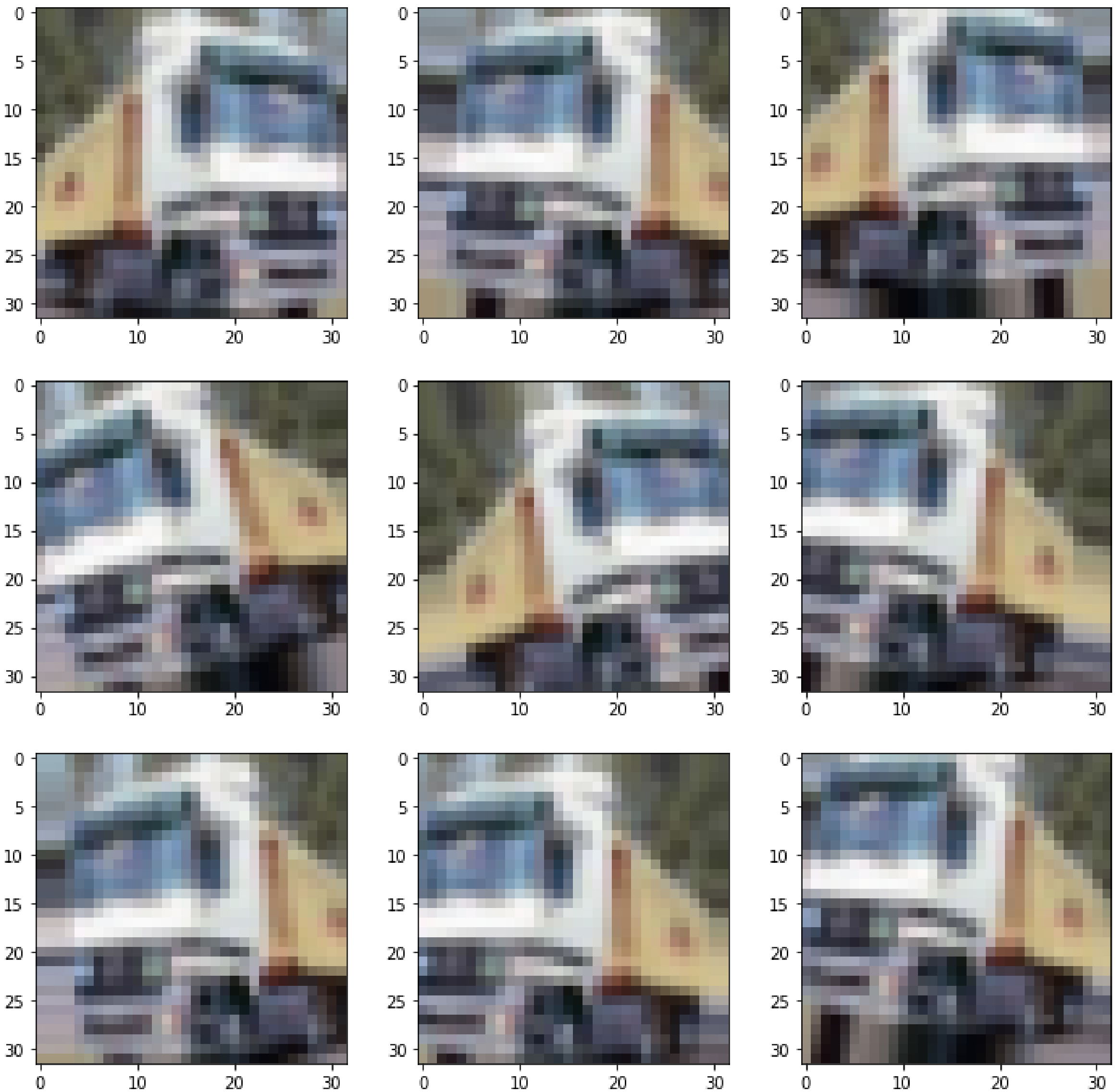
An example – pattern recognition with KERAS and TensorFlow

- CIFAR10 small image classification. Dataset of 50,000 32x32 color training images, labeled over 10 categories, and 10,000 test images.



https://github.com/marcinwolter/MachineLearning2020/blob/main/CNN_with_Image_Augmentation.ipynb

Data augmentation



Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

It acts as a regularizer and helps reduce overfitting when training a machine learning model.

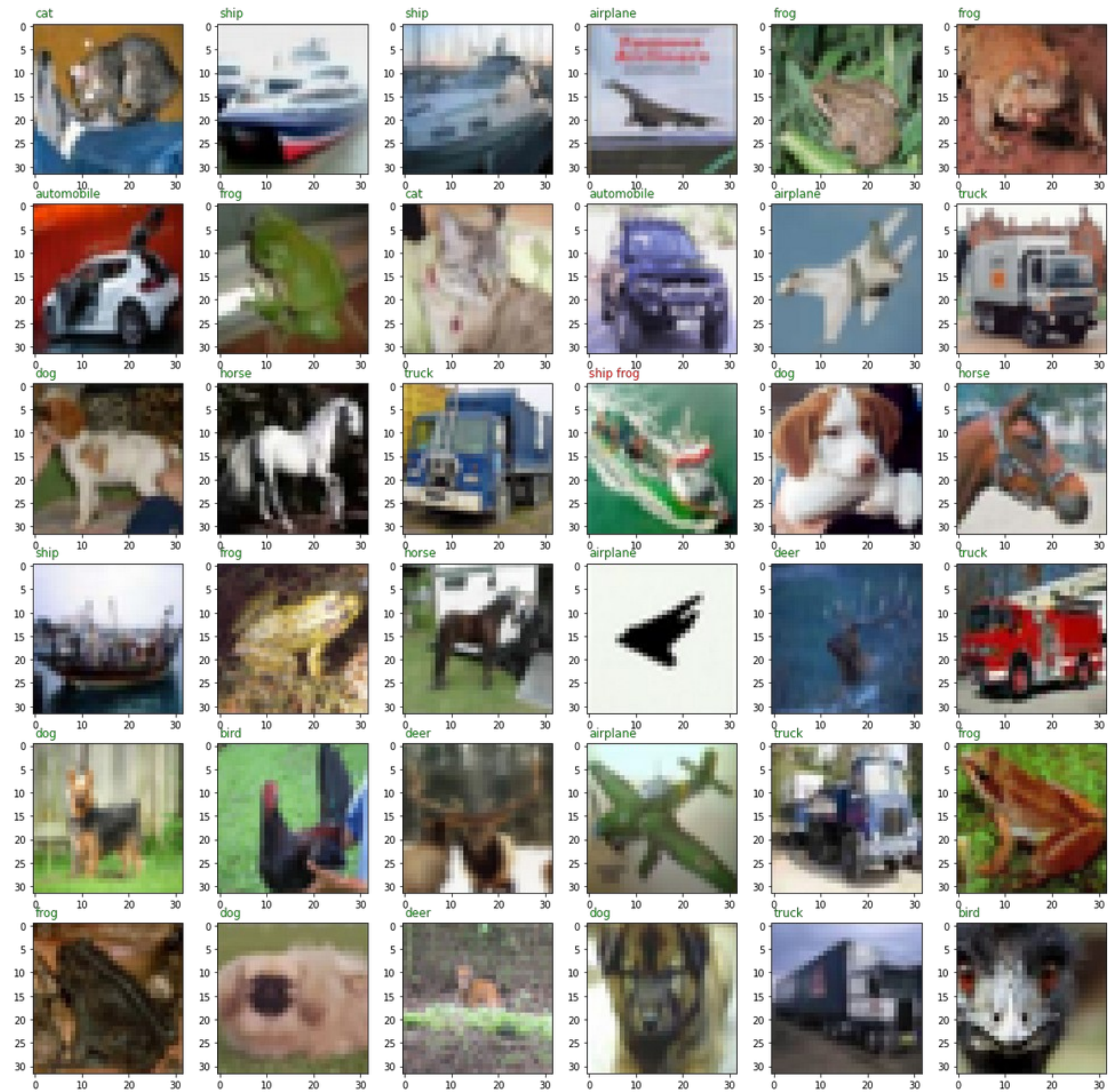
- Augmentation performed by:
  - Flip
  - Rotation
  - Shift

Deep Neural Network

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_18 (Conv2D)           (None, 32, 32, 32)        896       
_________________________________________________________________
batch_normalization_21 (Batc (None, 32, 32, 32)        128       
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 32, 32, 32)        9248      
_________________________________________________________________
batch_normalization_22 (Batc (None, 32, 32, 32)        128       
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
dropout_12 (Dropout)         (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 64)        18496     
_________________________________________________________________
batch_normalization_23 (Batc (None, 16, 16, 64)        256       
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 16, 16, 64)        36928     
_________________________________________________________________
batch_normalization_24 (Batc (None, 16, 16, 64)        256       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 8, 8, 64)          0         
_________________________________________________________________
dropout_13 (Dropout)         (None, 8, 8, 64)          0         
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 8, 8, 128)         73856     
_________________________________________________________________
batch_normalization_25 (Batc (None, 8, 8, 128)         512       
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 8, 8, 128)         147584    
_________________________________________________________________
batch_normalization_26 (Batc (None, 8, 8, 128)         512       
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 4, 4, 128)         0         
_________________________________________________________________
dropout_14 (Dropout)         (None, 4, 4, 128)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 512)               1049088   
_________________________________________________________________
batch_normalization_27 (Batc (None, 512)               2048      
_________________________________________________________________
dropout_15 (Dropout)         (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 10)                5130      
=================================================================
Total params: 1,345,066
Trainable params: 1,343,146
Non-trainable params: 1,920
_________________________________________________________________
```

Results

- About 90% accuracy (with 10 classes of images)



Summary

So, we know how to recognize what is on images!