TX00DH43-3001
Introduction to Deep Learning

peter.hjort@metropolia.fi

Computer vision

Typical problems in computer vision deal with:

Image classification: classify the image to one (or more) known categories. For example ImageNet has the images classified to 1000 categories (and some have multiple subcategories etc). (By the way: is 1000 much or little or just right?)

Object detection: detect interesting objects from a given image, and, for example, draw bounding boxes and classify them.

Image classification

We have already done this, what's the problem?

If we use a dense MNIST example network to classify into 10 categories:

- 64x64x3 image: 1,241,025 trainable parameters
- 256x256x3: 19,666,460
- 1024x1024x3: 314,578,460

(And this was with a small model)

Dense network and image classification

In a dense network everything is connected to everything.

But does that make sense when dealing with images?

- Perhaps images have more **structure** than just pixels → hierarchy of features
- Perhaps it makes sense to take a **more local look** at pictures → some features of the image might get repeated in other locations
- Perhaps we just can’t afford fully connected layers

Image classification and hierarchy

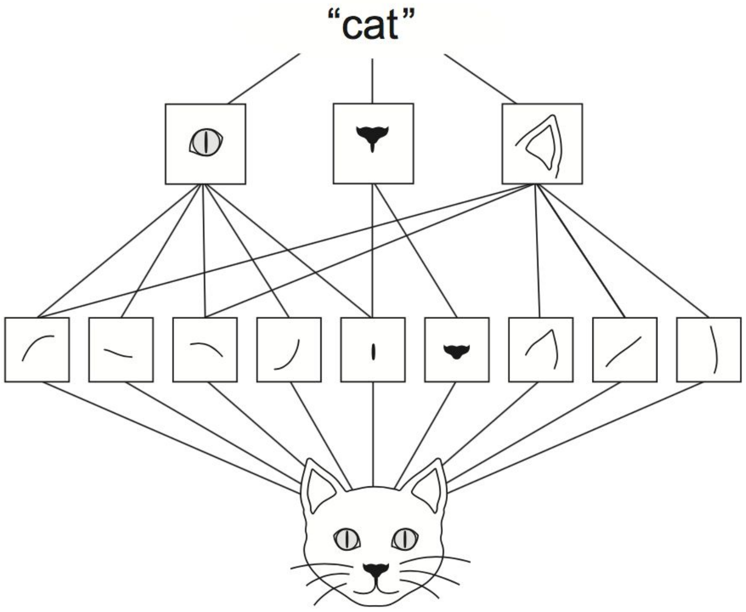
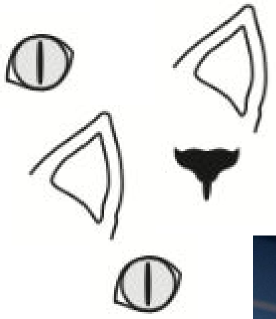


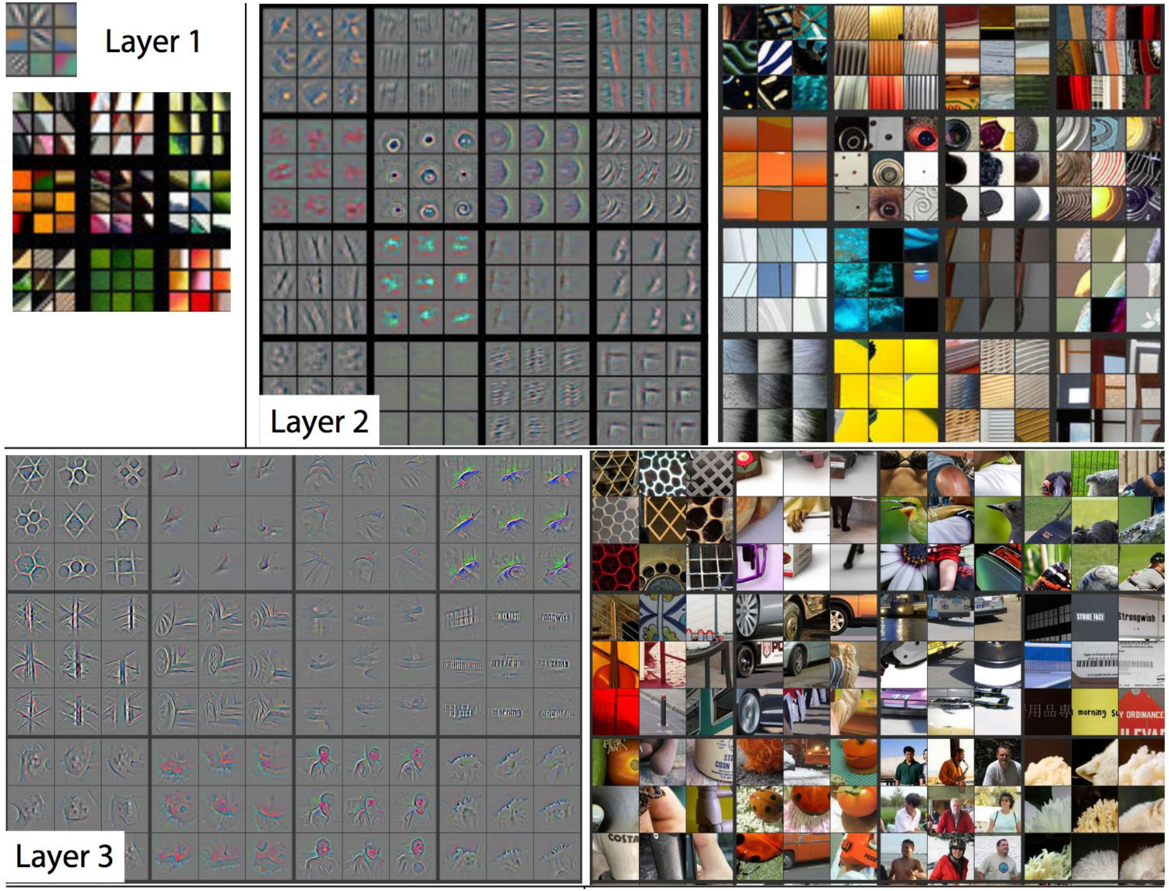
Figure 5.2 The visual world forms a spatial hierarchy of visual modules: hyperlocal edges combine into local objects such as eyes or ears, which combine into high-level concepts such as "cat."

From Chollet, chapter 5.1.1



Is this a cat ?

Image classification and hierarchy with filters



From Zeiler and Fergus “Visualizing and Understanding Convolutional Networks” (https://arxiv.org/abs/1311.2901)

2D convolution

Element-wise multiplication:
1*0+1*3+1*6+0*0+0*4+0*7+(-1)*5+(-1)*1+(-1)*1

0	0	5	2	1	4
3	4	1	1	1	3
6	7	1	1	1	4
8	9	2	1	5	2
2	0	0	0	2	1
6	2	2	2	2	1

6x6x1 image

*

1	0	-1
1	0	-1
1	0	-1

3x3 **filter**, or **kernel**

Filter size is sometimes called *receptive field*

2	7	4	-7
13	17	-3	-6
13	14	-5	-5
12	8	-5	-1

(6-3+1) x (6-3+1) x 1 result

What is happening in 2D convolution

9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0
9	9	9	9	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

0	0	27	27	0	0
0	0	27	27	0	0
0	0	27	27	0	0
0	0	27	27	0	0
0	0	27	27	0	0
0	0	27	27	0	0

This particular convolution filter/kernel seems to detect vertical edges.

Padding

Convolution shrinks the height and width dimensions of tensor. If no shrinking is desired, tensor can be padded with zeros before convolution.

- No padding: `'valid'` in Keras
- Pad to make input & output dimensions same: `'same'` in Keras
- Other values are possible, too (but not used much)

padding = 1

0	0	0	0	0	0
0	2	4	2	4	0
0	1	1	2	1	0
0	1	4	3	3	0
0	3	2	1	2	0
0	0	0	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

-5	-1	0	4
-9	-3	2	7
-7	-1	1	6
-6	0	1	4

Stride

of positions the convolution filter moves at one step.

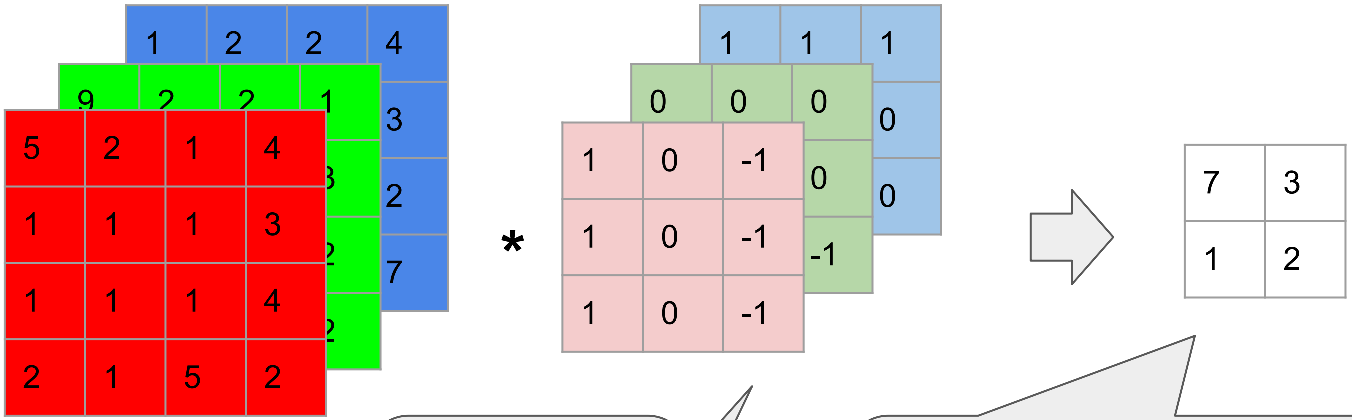
- Previous examples had stride = 1, so we were moving in one step at a time
- Stride = 2 - move 2 positions at a time:

$$\begin{bmatrix} 5 & -1 & 0 & 4 \\ -9 & -3 & 2 & 7 \\ -1 & -1 & 1 & 6 \\ -6 & 0 & 1 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} -4 & 2 \\ -7 & 2 \end{bmatrix}$$

- Strides are more often used in pooling operations (wait for a couple of slides)

RGB image convolution

Think of RGB image as 3-dimensional box, where channels are in depth direction:

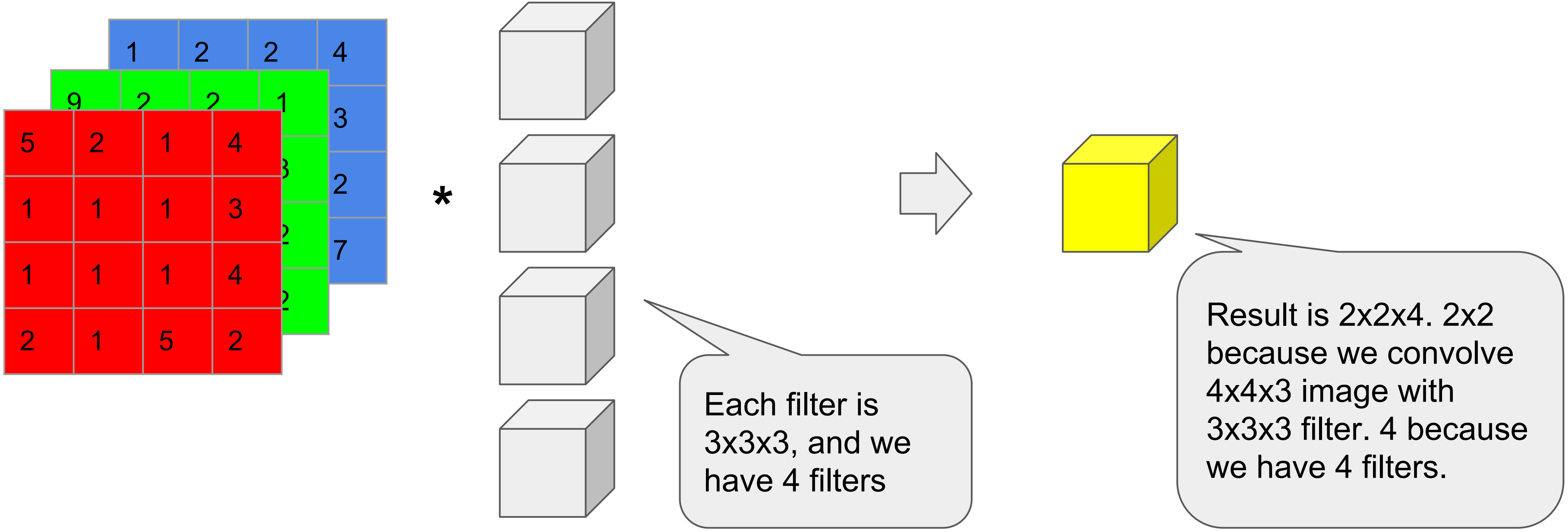


Filter must have the same depth as the sample.

Upper left corner 7 is the result of doing convolution in upper left corner of red channel with light red filter, green with light green filter, blue with light blue filter, and summing all.

Convolution with multiple filters

Usually more than one filter is used in convolution. Each filter is **applied separately**, and **results are stacked in depth dimension**:



Where do the convolution filter values come?

They are **learnable parameters** (or weights)!

We are not designing the filters beforehand, like in signal processing, but **learn the values in filter during training**. This gives the network ability to **learn whatever filters to minimize the loss function**.

Filter parameters get learned once, and are then used in all spatial positions (height & width) of the sample image - parameter sharing.

of parameters/weights in convolutional layer

```
model.add(Conv2D(filters=6,
                 kernel_size=5,
                 strides=1,
                 padding='valid',
                 input_shape=(28, 28, 3,),
                 activation='relu'))
```

((5 x 5 x 3) + 1) * 6 = 456 (450 if `use_bias` parameter in `Conv2D` is set to `False`)

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 6)         456
_________________________________________________________________
```

Max pooling

2	7	4	-7
13	17	-3	-6
13	14	-5	-5
12	8	-5	-1

Filter size = 2
Stride = 2

17	4
14	-1

“Prominent features”

Another option: average pooling (not very common)

Max pooling is done independently for all channels.
Note: no parameters to learn.

Typical convolution network structure

One or more groups of one or more convolution layers, followed by pooling layer

```
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(28,28,1))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Group of dense layers, followed by softmax output layer

```
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

Flattening layer

In the previous example the output of `MaxPooling2D` layer has shape `(12,12,64)`. How do we feed this to a `Dense` layer?

Flatten it:

```
model.add(Flatten())
```

This will output shape `(9216)` (12*12*64)

Data augmentation

Too few samples to train the model?

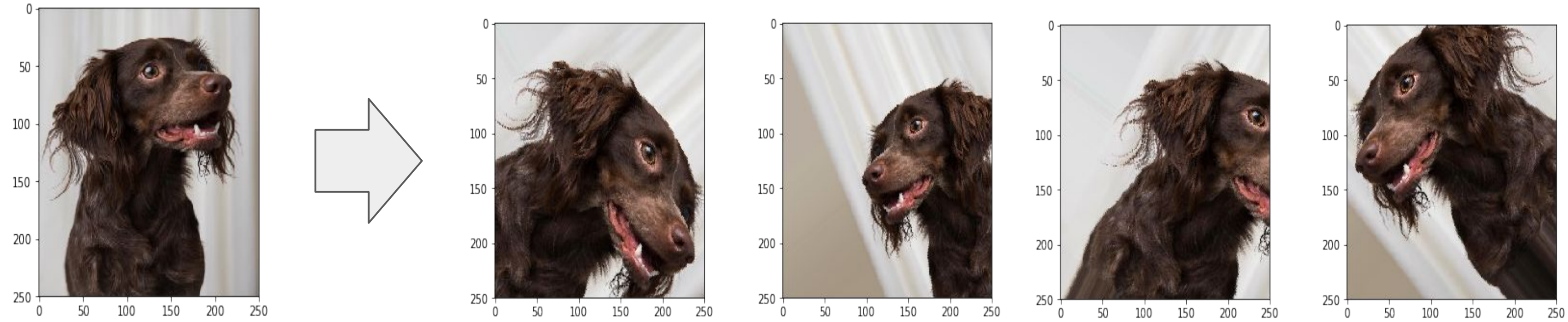
Don’t worry - let’s create fake data! Well not completely fake, but for image data, for example, data from real images by random modifications:

- Rotation
- Zooming
- Flipping
- (and more, see https://keras.io/preprocessing/image/ and Chollet 5.2.5)

Data augmentation with `ImageDataGenerator`

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
      rotation_range=60,
      zoom_range=0.2,
      width_shift_range=0.4,
      height_shift_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```



Preprocessing and generators

Keras preprocessing.image has ImageDataGenerator that can be used for:

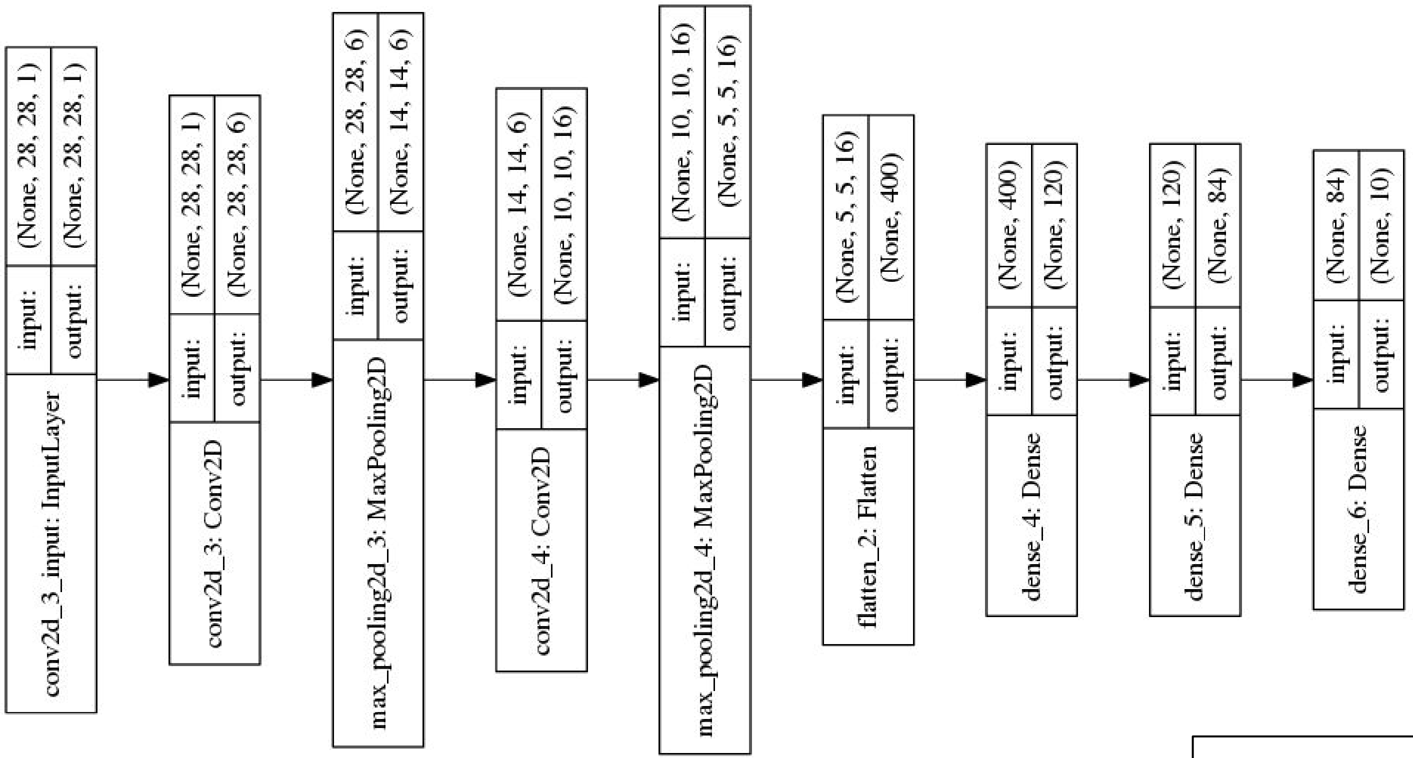
- Reading from disk and preprocessing data in batches
- Making transformations to augment data

```
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_dir,
                          target_size=(150, 150)
                          batch_size=20,
                          class_mode='binary')
```

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=30,
      validation_data=validation_generator,
      validation_steps=50)
```

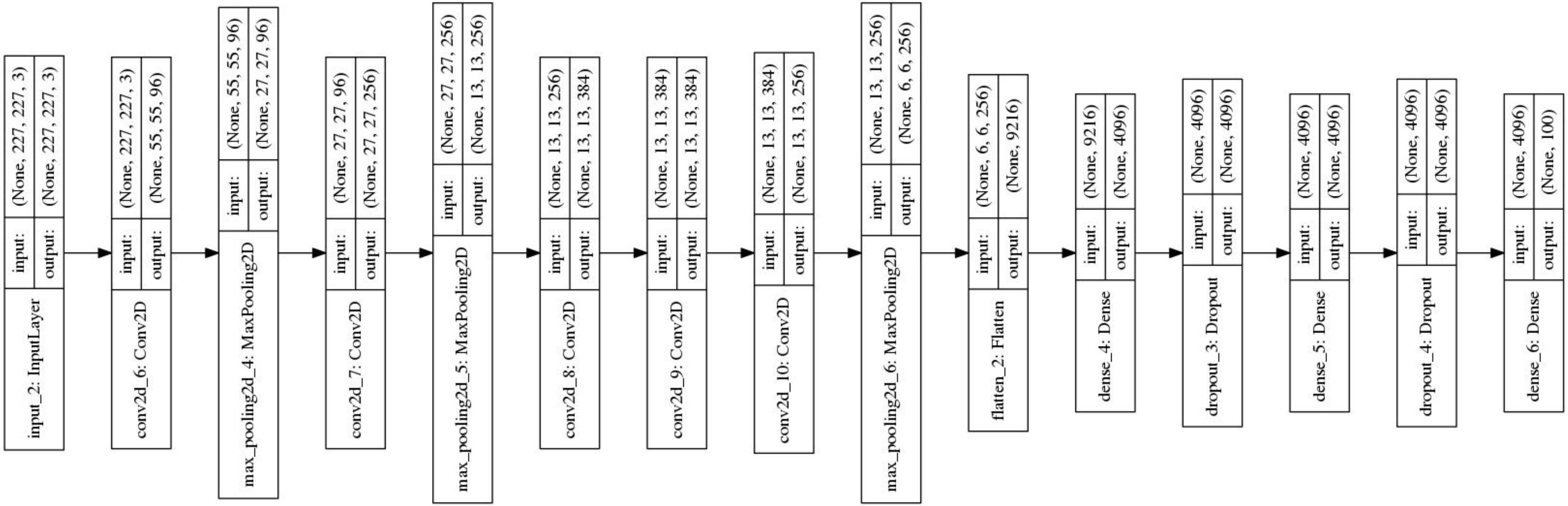
One epoch is
20 * 100
training
samples

Example network: LeNet-5 (1998) (simplified)



http://yann.lecun.com/exdb/lenet/

Example network: AlexNet (2012) (simplified)



https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

VGG-16 (2015)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

From https://arxiv.org/pdf/1409.1556.pdf

References

"Visualizing and Understanding Convolutional Networks" by Zeiler and Fergus.
https://arxiv.org/abs/1311.2901

How convolutional neural networks see the world.
https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Reading list for exam 12.2

session04.pdf

Chollet: 5.1, 5.2, 5.4 (code examples in 5.4 can be ignored, concentrate on the convnet visualizations and heatmaps)

Exercise: fashion MNIST with convolutional network

Use a convolutional network with overall structure:

- (1-2 * convolution layers; pooling layer) * 1-3
- dense layer * 1-2
- softmax

to train a network to classify fashion MNIST data set. Analyse model performance; can you beat your dense network performance?

Exercise*: Cifar-10

Train a convolutional network inspired by AlexNet structure to classify Cifar-10 data set. Use `ImageDataGenerator`. Note: start with small network, watch out # of parameters. Is Cifar-10 harder/easier to train than fashion MNIST? Any other findings?

Note: you can also train a model with less than 10 categories if you run out of computational power or patience. (Some manipulation of the data set is needed.)

Dataset is available in keras.datasets. For information about Cifar-10, see for example https://www.cs.toronto.edu/~kriz/cifar.html

*) This exercise is not mandatory.

Exercise*: handwritten digits and data augmentation

Can you improve classification accuracy of real handwritten images (*Exercise: train and use MNIST* in session02) by using a convnet model and data augmentation?

*) This exercise is not mandatory.