Lab 12. VGGNet

Intro to Machine Learning
Fall 2018, Innopolis University

Lecture recap

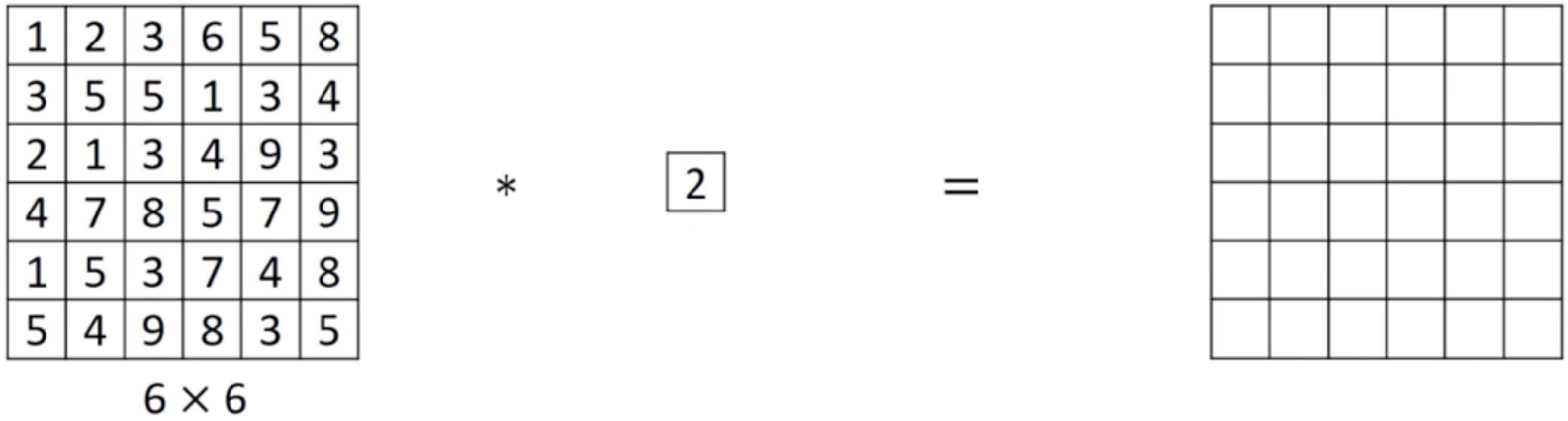
What are the differences between architectures?

- LeNet
- ImageNet
- AlexNet
- ZFNet
- GoogleNet
- ResNet

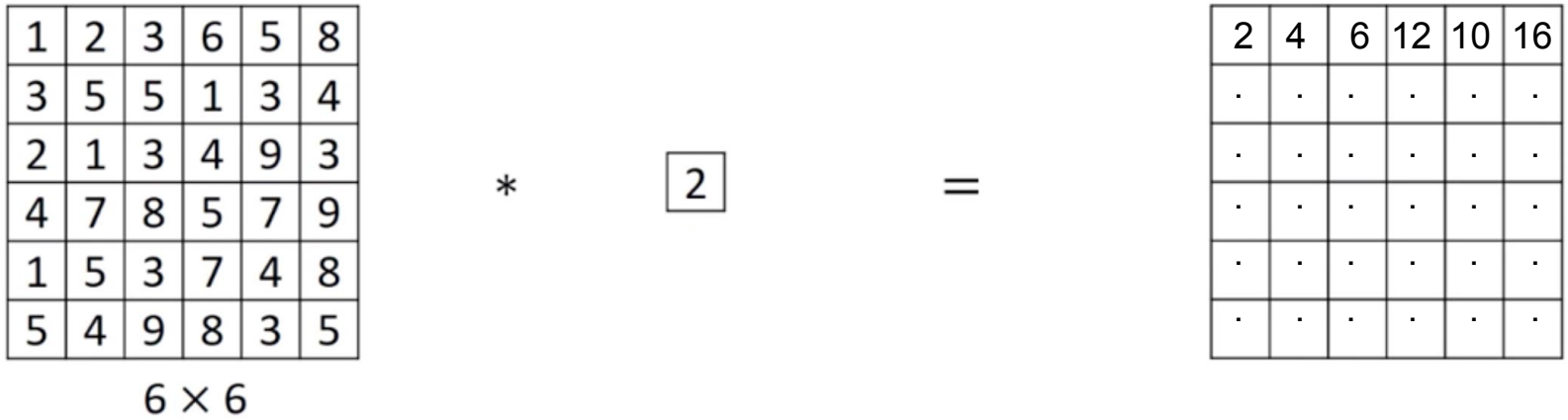
Today's plan

1. 1x1 convolutions
2. Mini VGGNet implementation
3. Data augmentation
4. Dropout
5. Homework explanation

What does 1x1 convolution do?

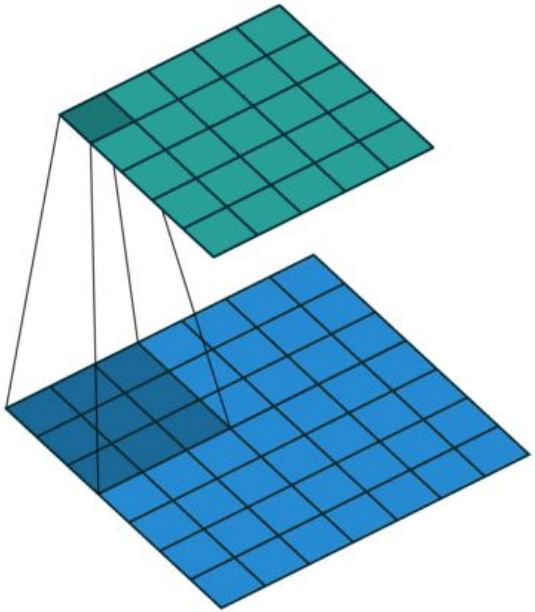


What does 1x1 convolution do?

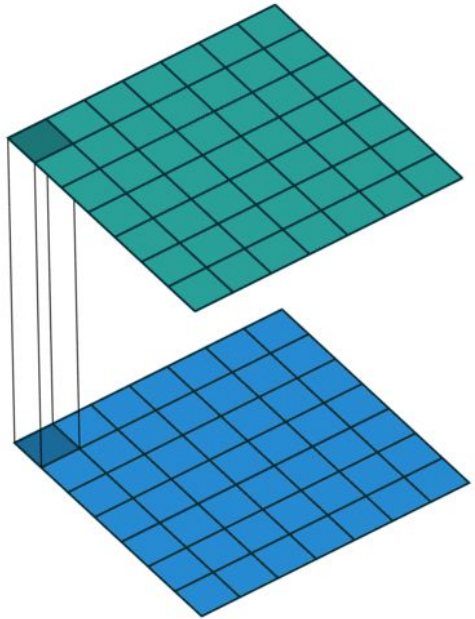


It is a simple multiplication by a number

What does 1x1 convolution do?

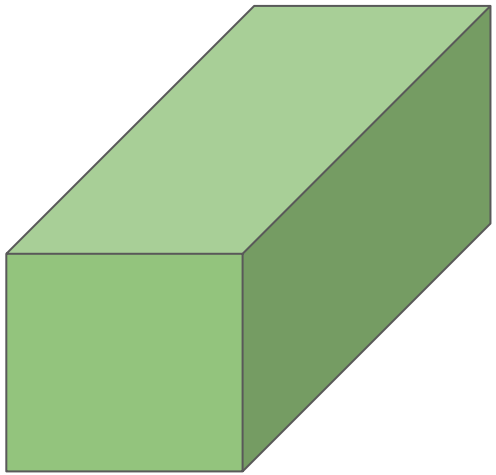


Kernel of size 3x3

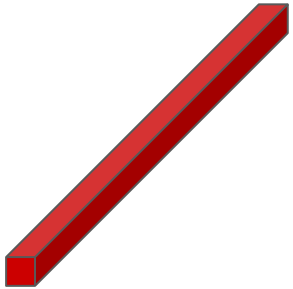


Kernel of size 1x1

What does 1x1 convolution do?



*



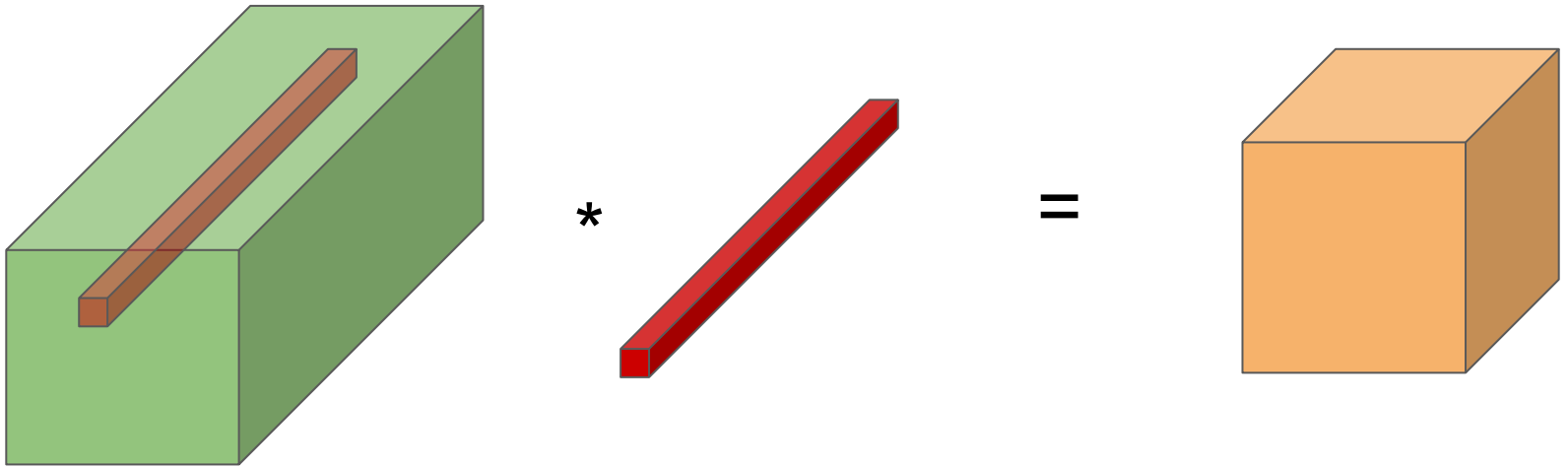
=

?

6 x 6 x 32

1 x 1 x 32 x number of filters

What does 1x1 convolution do?



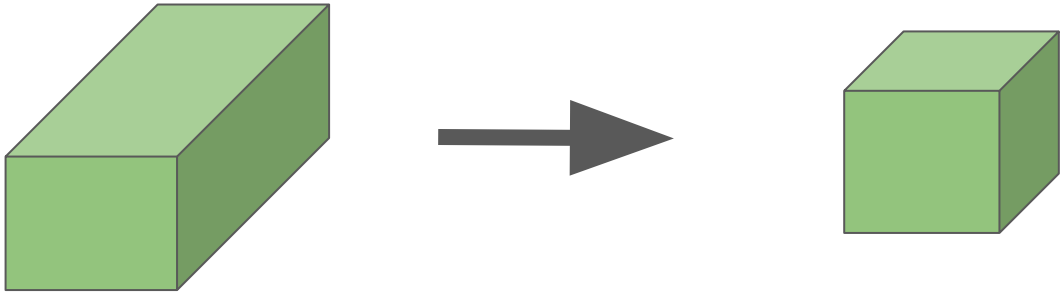
6 x 6 x 32

1 x 1 x 32 x number of filters

6 x 6 x number of filters

Using 1x1 convolutions as “feature pooling”

1 x 1 convolution



28 x 28 x 192

28 x 28 x 32

pooling

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

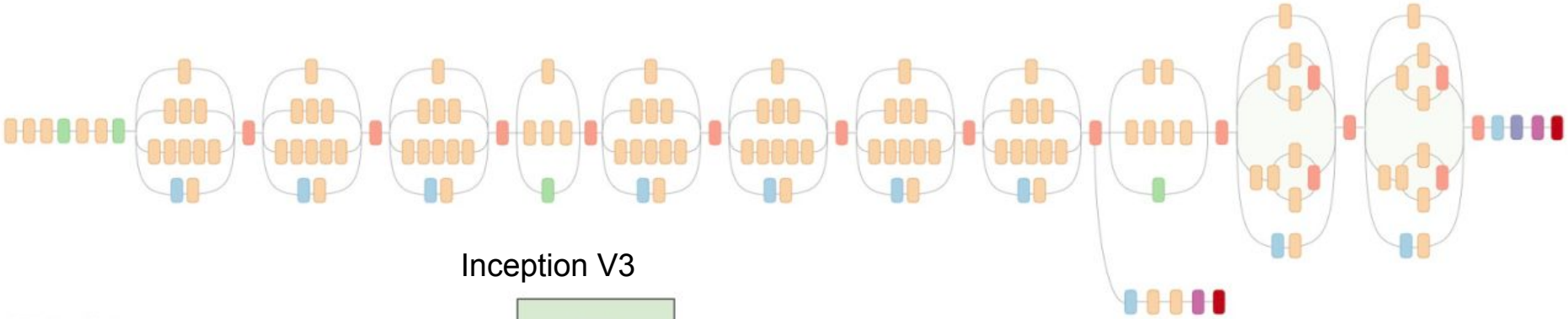
4 x 4 x 1

2×2 Max-Pool

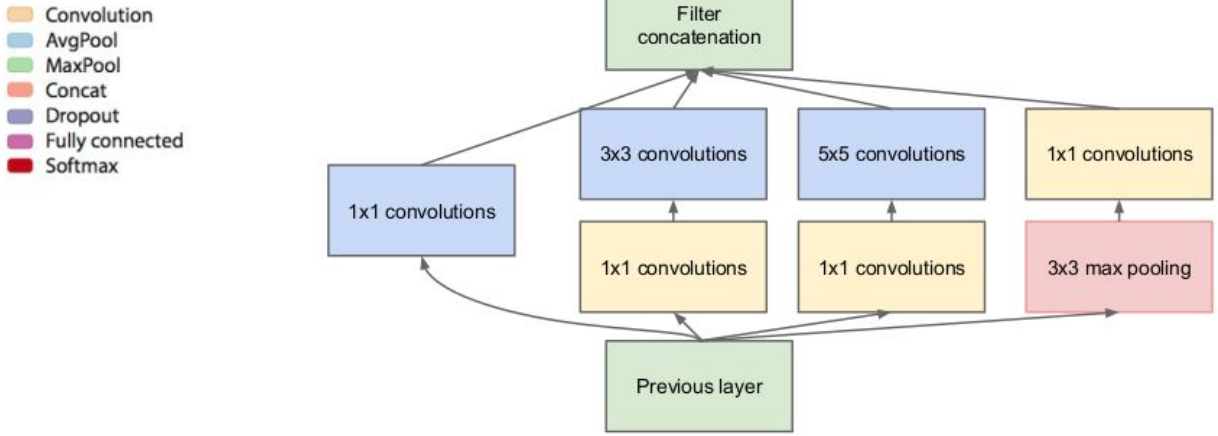
20	30
112	37

2 x 2 x 1

Using 1x1 convolutions

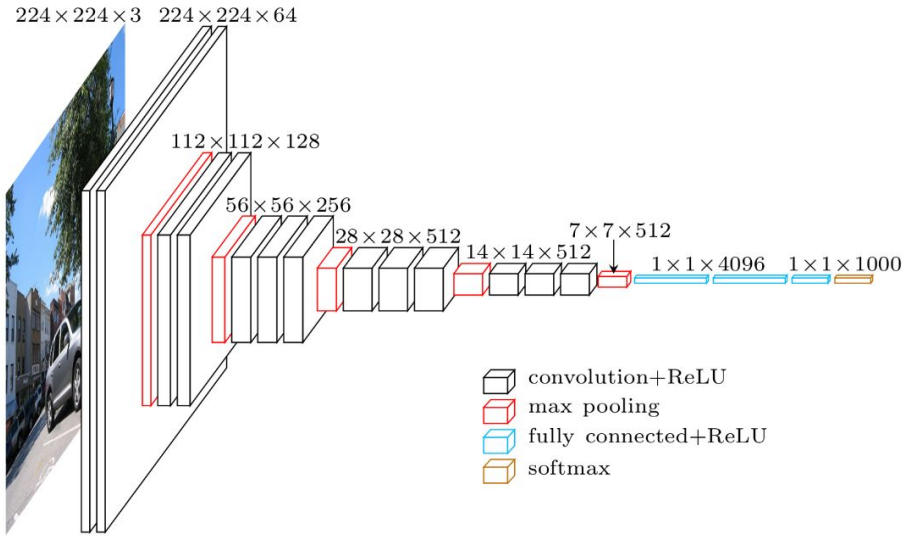


Inception V3



Inception module

VGGNet



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					

VGG16 Keras implementation

```
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(224,224,3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Conv2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1000, activation='softmax'))
```

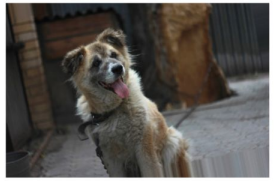
OR

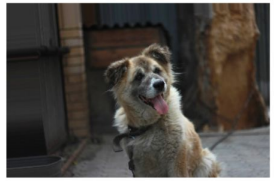
```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
# load the model
model = VGG16()
# load an image from file
image = load_img('cat.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

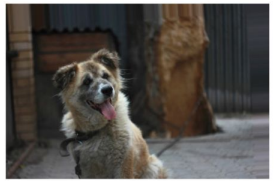
Data augmentation

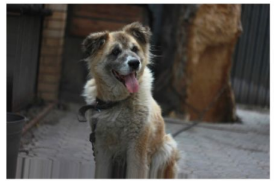
```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import array_to_img, img_to_array, load_img

input_path = 'dog.jpg'
output_path = 'dog_aug{}.jpg'
count = 9

gen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

# load image to array
image = img_to_array(load_img(input_path))

# reshape to array rank 4
image = image.reshape((1,) + image.shape)

# let's create infinite flow of images
images_flow = gen.flow(image, batch_size=1)
for i, new_images in enumerate(images_flow):
    i+=1
    # we access only first image because of batch_size=1
    new_image = array_to_img(new_images[0], scale=True)
    new_image.save(output_path.format(i))
    if i >= count:
        break
```

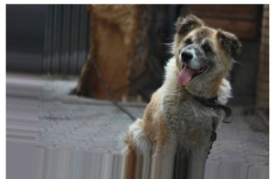
Data augmentation result

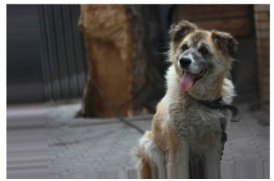




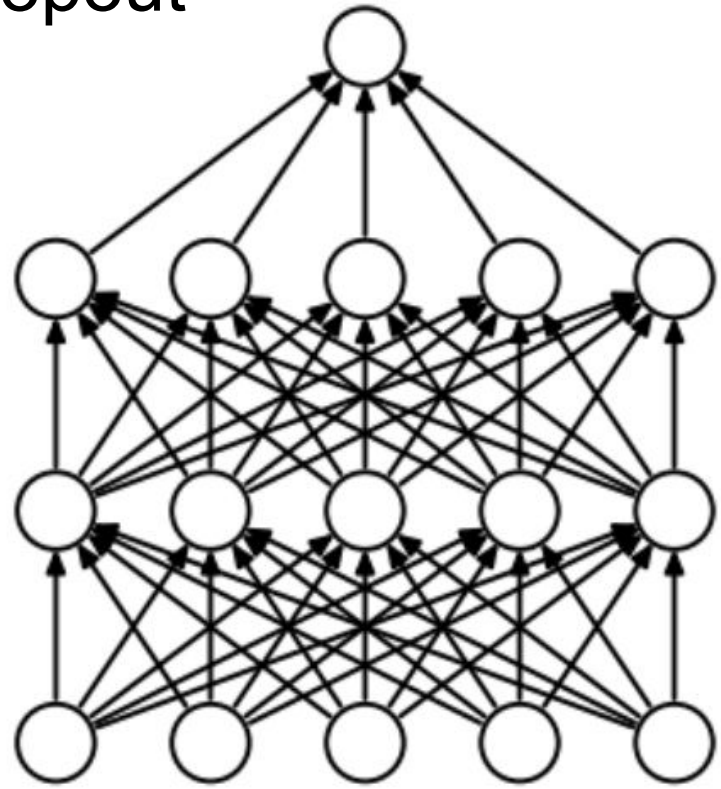




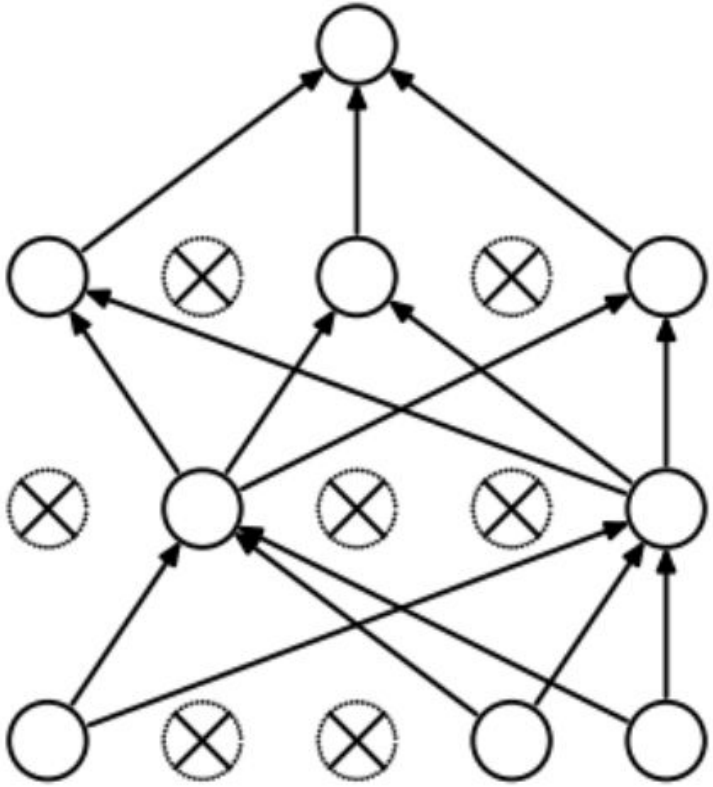




Dropout



(a) Standard Neural Net



(b) After applying dropout.

Dropout in Keras

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

References

1) Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).
2) Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
3) https://iamaaditya.github.io/2016/03/one-by-one-convolution/
4) https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Homework explanation

Read GoogleNet and ResNet original papers and write a report in your own words (1 page each) **No copy paste allowed!** You should include:

- Key contribution of the paper

- Key differences with the state of the art: contrast this method against others (commonly can be extracted from the intro or related works)

- Explain the method in your own words. If the paper contain mathematical results, you should include these as well, explaining each term.

- Key results: express the good, the bad, and the ugly of the results. Explain standard benchmarks and results on them.

Homework explanation

Again, **copy paste**.

That’s it for today! Questions?