MASTER OF SCIENCE
IN ENGINEERING

Typical Exam Questions

TSM_DeLearn

Some are rather sketchy

Jean Hennebert
Martin Melchior

General guidelines to prepare for the exam

- study the slides and additional material we gave (pointers, videos, etc)
- be able to redo the PW including interpretation of obtained results
- go through the review questions provided with each PW
- go through this document that explains typical exam questions
- take part to the Q&A session on 23rd of January

Type of Questions

- **Targeting theory and concept:** Focus on definitions given in class (blue frames), understanding concepts such as loss function, gradient descent, optimisation/regularisation procedures, computational graphs, “static” graph strategy or eager execution in TensorFlow, functioning of layers composing a CNN, intuition behind deep architectures, etc.

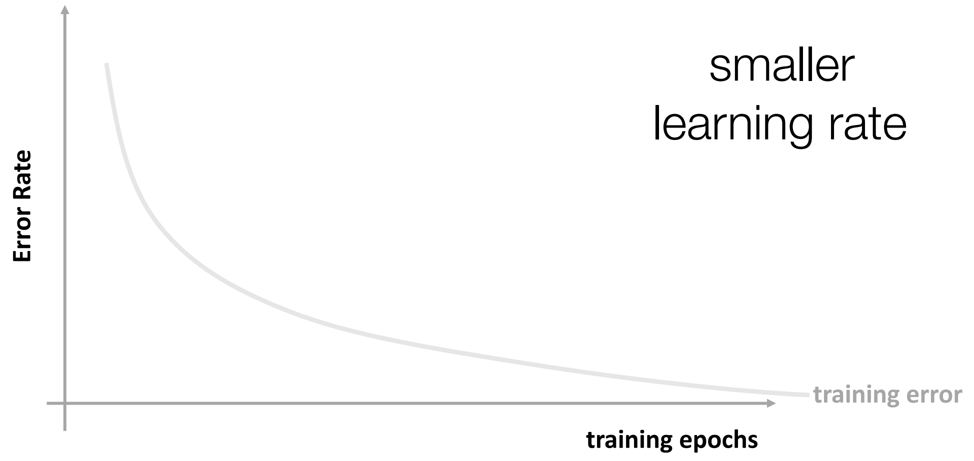
- **Pen-paper exercise:** E.g. computing the numerical values of gradients in a computational graph, computing the activation map on a simple example, computing the number of parameters of a given architecture.

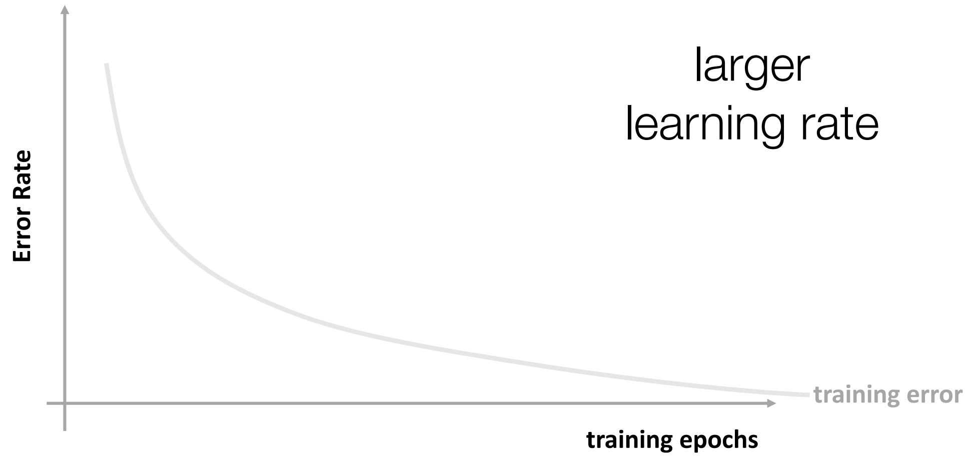
- **Code reading:** Re-explaining a piece of code that you had to produce during the practical works, eventually filling holes in a proposed code (we will not penalise syntax error, we want to see concept such as: “here I would add a Dense layer…”)

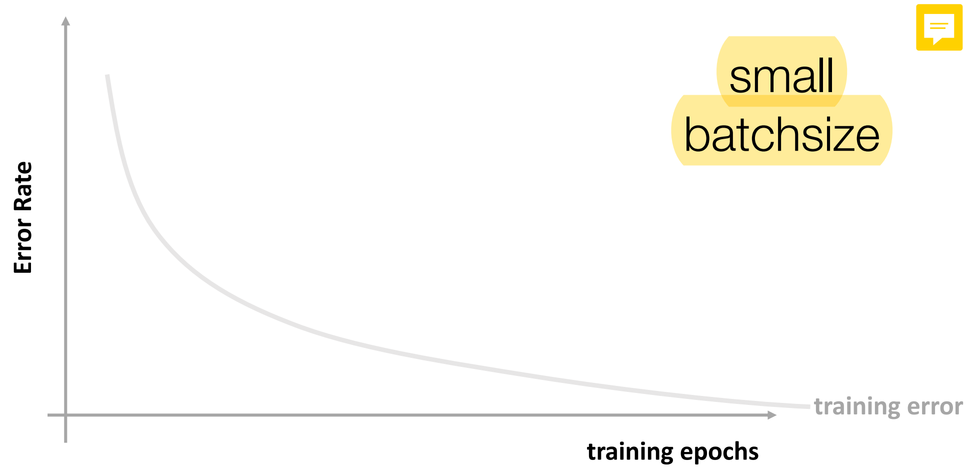
- **Use case questions:** In front of a given real-life problem described in, let’s say half a page, what would be your deep learning strategy

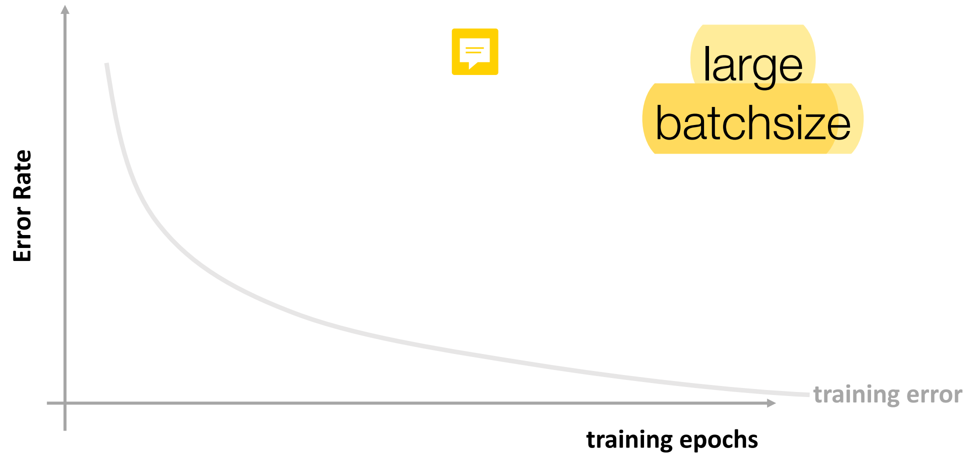
Learning Curves (1)

- Plot qualitatively the changes you expect when modifying the indicated hyper-parameter in the given direction. Express that in 1-2 sentences and explain why it changes the way you expect. The given grey line provides a smoothed baseline.





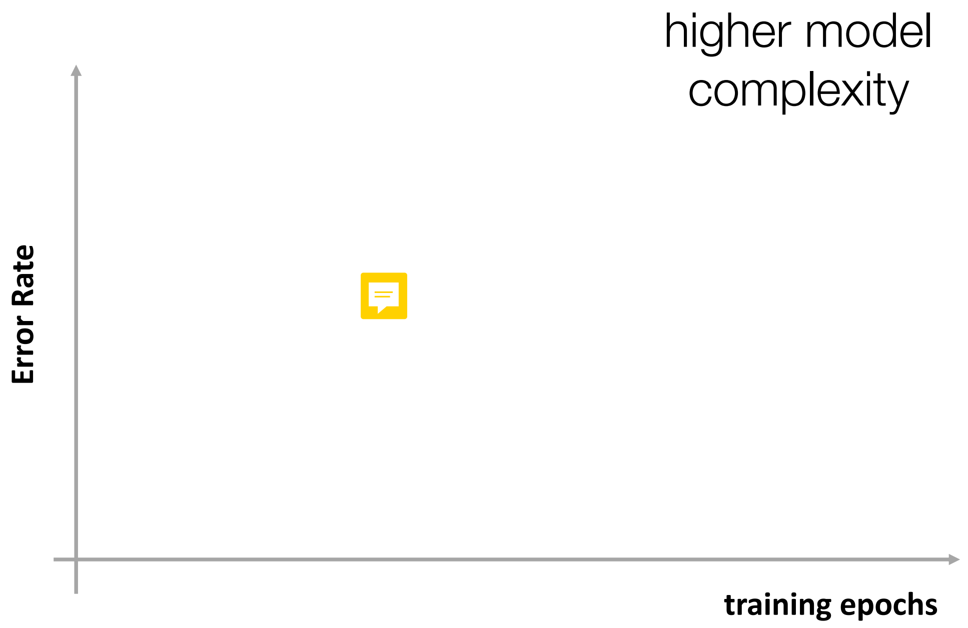




Learning Curves (2)

- Plot qualitatively the train and test error rates you expect for the given change in the settings (to be compared horizontally). Express that in 1-2 sentences and explain why it changes the way you expect.

lower model complexity
Error Rate
training epochs

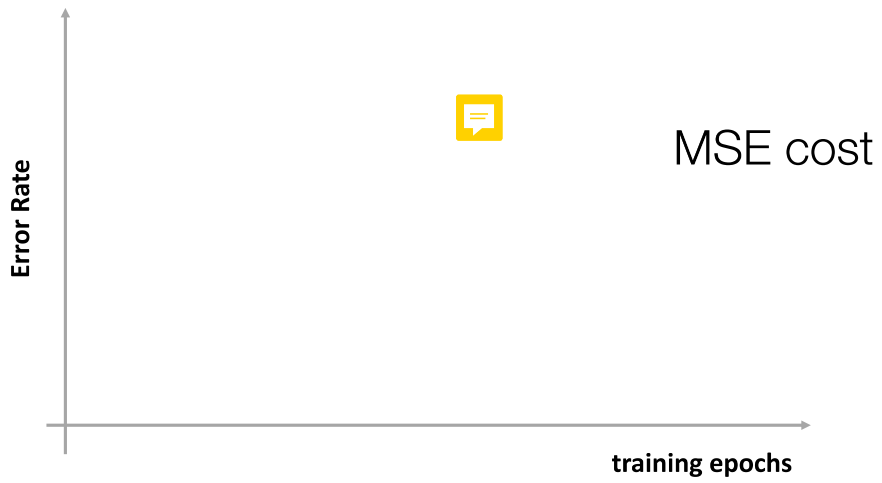


- What change do you expect to see when the parameters (weights) are not properly initialised?

Learning Curves (3)

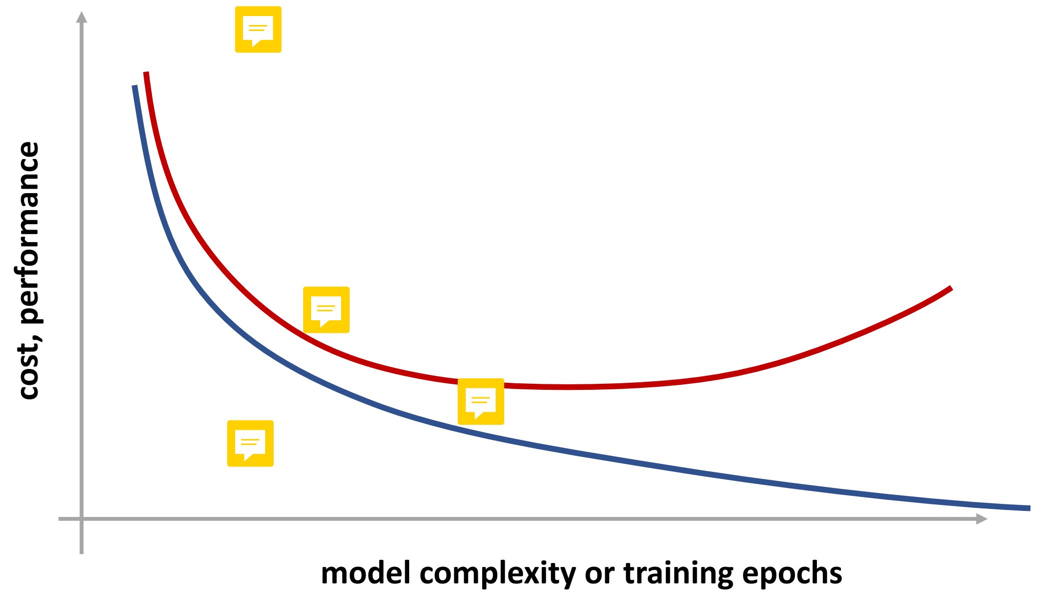
- Consider the situation of training a model for a classification task. Plot qualitatively the train and test error rates you expect for the given change in the settings (to be compared horizontally). Express that in 1-2 sentences and explain why it changes the way you expect.

Error Rate
cross entropy cost
training epochs



Learning Curves (4)

- Identify overfitting/underfitting from given learning curves
  - In the plots identify the quantities: variance error, bias and generalisation error, test error, training error.
  - Describe the means to make them smaller.



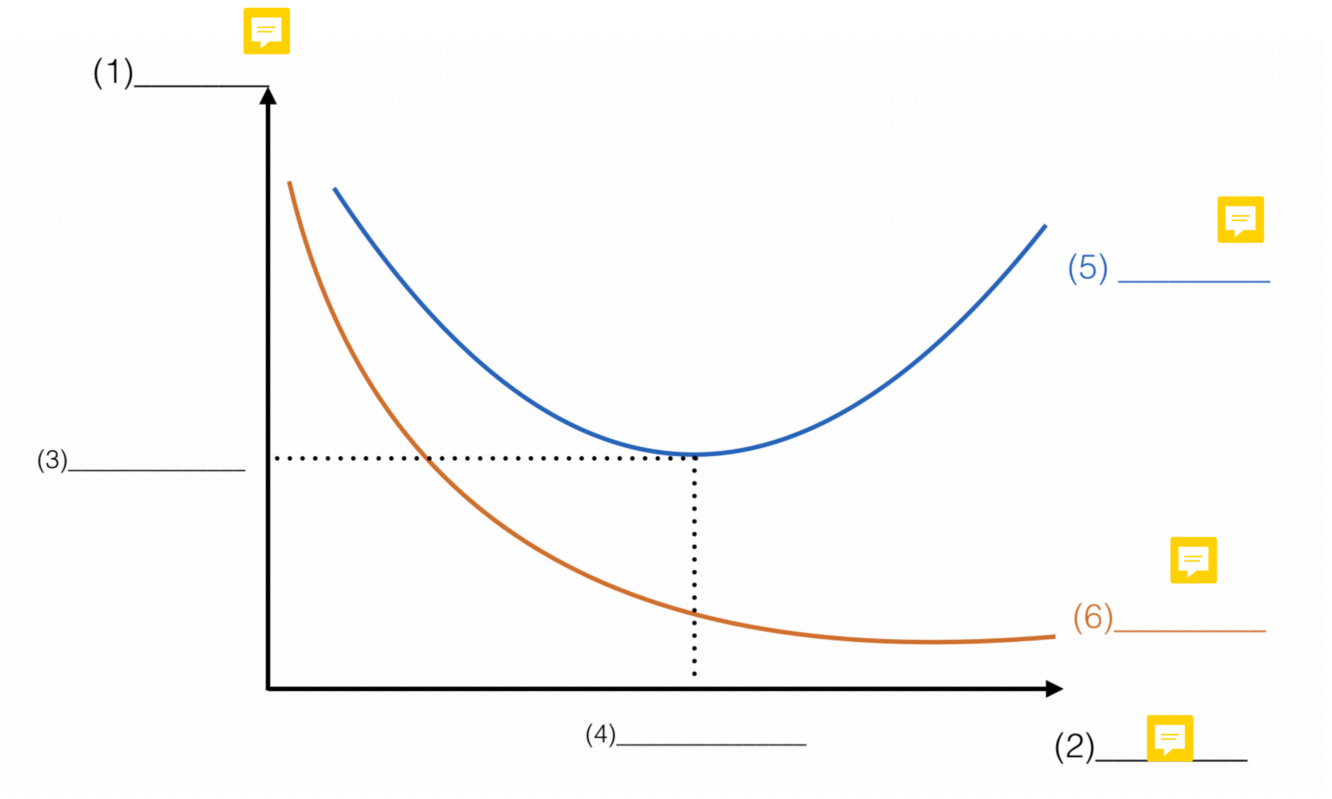
- Explain what effect the operations listed in the table generally have on the bias and variance of your model. Fill in one of 'increases', 'decreases' or 'no change' in each of the cells.

	Bias	Variance
Regularizing the weights		
Increasing the size of the layers (more hidden units per layer)		
Using dropout to train a deep neural network		
Getting more training data (from the same distribution as before)		

Learning Curves (5)

- Explain how the training and the validation/test cost change with increasing number of samples.

- Explain the concept of early stopping with the figure on the r.h.s. Fill-in the blanks: (1) and (2) describe the axes, (3) and (4) describe values on the vertical and horizontal axis, (5) and (6) describe the curves. Be precise.



Gradient Descent/Optimisation (1)

- Cost Function
  - Explain what the cost function is used for. Specify (incl. formulas) the typical costs for
    - a regression task
    - a binary classification task
    - a multi-class classification task
  - Formulate the explicit update step for a linear regression problem. Then add L2 regularisation and compute the modified update rule. What will be effect of this regularisation?
    [Here, we probably would be a bit more specific]
  - Explain why cross entropy is better suited for classification tasks.
  - Compute the cross entropy loss for given labels y=1,3,2 and model scores

$$\hat{\mathbf{y}} = \begin{pmatrix} 0.4 \\ 0.3 \\ 0.3 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.5 \\ 0.2 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.5 \\ 0.2 \end{pmatrix}$$

- Gradient Descent
  - Describe applicability and convergence properties of gradient descent
  - Explain why gradient descent is considered as a ‘local’ learning principle and what the implications of this are.
  - Describe characteristics and pro’s/con’s of batch gradient descent, mini-batch gradient descent, stochastic gradient descent.

Gradient Descent/Optimisation (2)

- Mini-Batch Gradient Descent
  - Explain why mini-batch gradient descent is considered as preferred over batch or stochastic gradient descent.
  - Write down pseudo-code for mini-batch gradient descent for given
    - model function `model(x,p)` with input data **`x`**, parameters **`p`**.
    - function for the gradient of the per sample loss w.r.t. the parameters **`p`**, **`gradient(p,x)`**
    - input dataset provided by a 2d numpy array with shape **`(n,m)`** with **`n`** input features and **`m`** samples.

General Machine Learning

- Draw a flow chart for a typical machine learning pipeline. Give a short description for each of the steps (1 sentence each) and explain in what situation (1 sentence each) and why each of the steps is important (1 sentence each).

- What's the risk with tuning hyper-parameters using a test dataset?

- Assume you design a model to predict the presence or absence of a tumor in a brain scan. The goal is to ultimately deploy the model to help doctors in hospitals.
  - What performance metrics do you use? Explain.

- Consider the confusion matrix
  - Compute the total number of samples. Are these samples of the training or test dataset?
  - Write down the confusion table for class A
  - Compute the overall accuracy and overall error rate of the system
  - Compute the per class recall for all three classes
  - Compute the per class precision for all three classes

	Apred	Bpred	Cpred
A	20	2	3
B	4	10	1
C	3	2	15

General machine learning

- On stochastic (and mini-batch gradient descent), what is the risk if the engineer forgets to shuffle the data set?
- Assuming a classification problem with 3 classes C1, C2 and C3 with a priori probabilities of the classes in the training and testing set equals to P(C1)=20%, P(C2)=30% and P(C3)=50%:
  - What is the lower bound of accuracy your system should have?
  - Assuming you deploy the trained model in a context where all classes have now equal a priori probabilities:
    - What problem can you expect?
    - How to solve this problem?

Representational Capacity

- Given several models for an image classification task (model complexity indicated) and given error rates: Indicate what error rates are produced by which model. Explain!

- Sketch a constructive proof for why shallow networks (with a single hidden layer and sigmoid activation function) can be used to arbitrarily well approximate continuous functions on finite intervals.

- Explain what the “universal approximation theorem” implies in view of using shallow networks. Explain the difficulties that occur when using only shallow networks and explain in what sense deep networks provide help.

Activation Functions

- Explain why non-linear activation functions are an important ingredient of neural networks.

- Describe pro's and con's of using relu vs sigmoid. Explain possible issues one needs to deal with when learning models with either of the two functions. Provide typical situations where the two are applied.

- Consider a neuron with sigmoid activation function and scalar input x, weight w and bias b, i.e. $\sigma(wx+b)$ When compared with w=1, b=0 (i.e. $\sigma(x)$) describe the impact of choosing
  - a negative bias
  - a large positive weight (e.g. w=10)
  - a small negative weight (e.g. w=-0.5)

- If we initialize the weights all equal to 0.0, would the network still converge to something?

Multi-Layer Perceptron (1)

- Compute the number of trainable parameters for a MLP with layer sizes 100,50,20 that processes images of size 20x20x3.

- Given a binary classification task for images of shape 64x64x3. The task is modelled with a network with a single hidden layer of 100 units. Compute how much RAM is needed to represent the trainable model parameters in memory (assume that each parameter is represented by float values of 64 bits.

- Compute the softmax values for the given matrix of logits (samples in the columns). Express the result with exponentials without summation symbol.

$$\mathbf{z} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 3 & 1 \\ 2 & 2 & 0 \end{pmatrix}$$

Multi-Layer Perceptron (2)

- Suppose you have a 3-dimensional input x= (x1,x2,x3)=(4,2,1) fully connected to 1 neuron with activation function gi. The forward propagation can be written:

$$z = \sum_{k=1}^{3} w_k x_k + b \ , \ a_i = g_i(z)$$

  After training this network, the values of the weights and bias are:
  w=(0.5,-0.1,0) and b=0.2.
  You try 4 different activation functions (g1,g2,g3,g4) with

$$gi \in \{\text{sigmoid}, \text{relu}, \text{heaviside}, \text{linear}\}$$

  that lead to activations (a1,a2,a3,a4) where $a_1 \leq a_2 \leq a_3 \leq a_4$. What is a valid guess for the activation functions gi ?

- Why do we need a bias?

- Explain why we need to randomly initialise the weights. Explain why the scale of the initial weights needs to be chosen carefully.

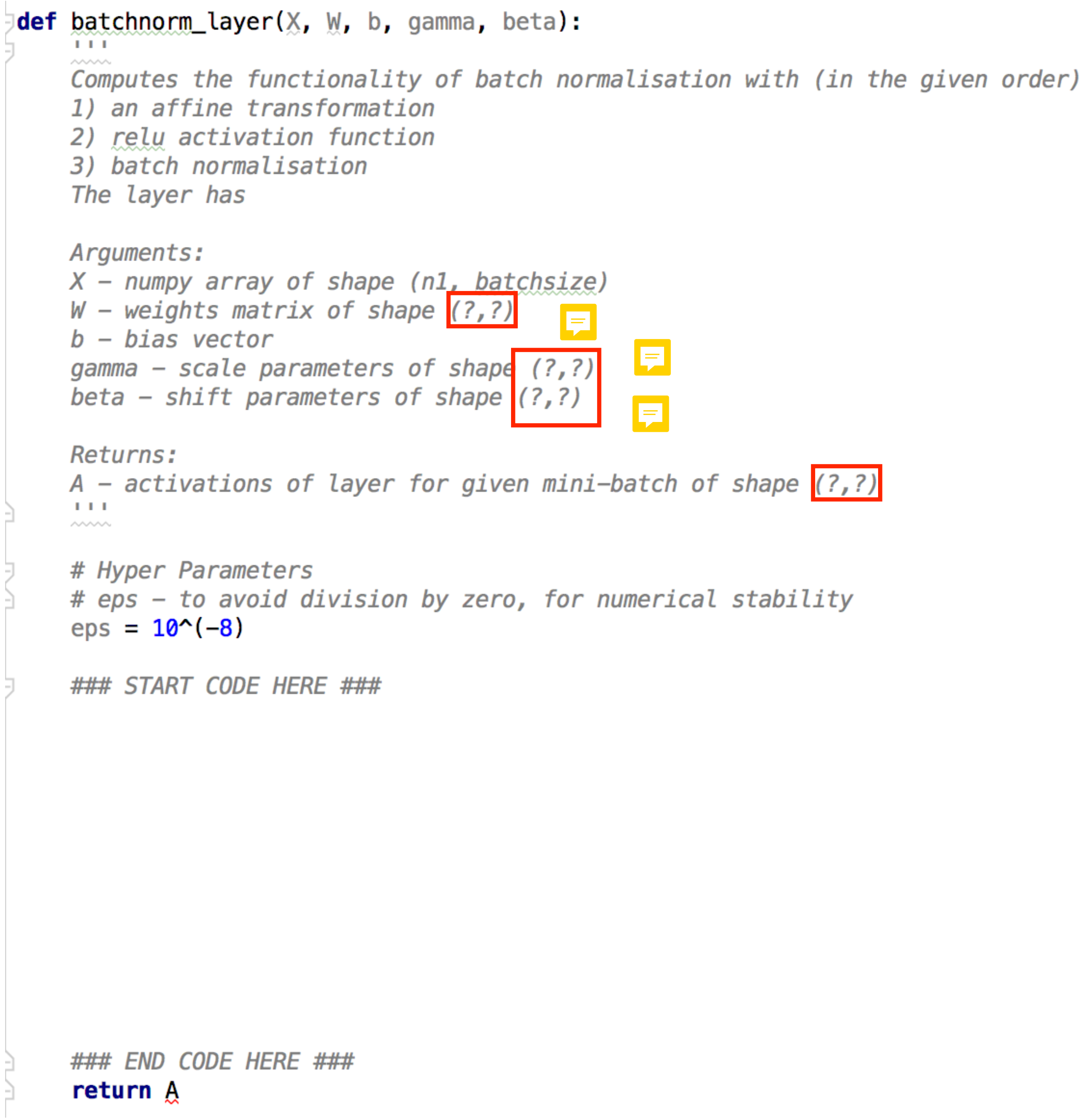
Multi-Layer Perceptron (3)

- Consider an input image of shape 500×500×3. You flatten this image and use a fully connected layer with 100 hidden units. What are the shapes of the weight matrix and the bias vector?

- Consider an input image of shape 500×500×3. You run this image in a convolutional layer with 10 filters, of kernel size 5×5. How many parameters does this layer have?

Batch-Normalisation (1)

Fill in the blanks — numpy code for computing a batch-normalised relu layer. Also specify the shape (marked with red circles)



```
def batchnorm_layer(X, W, b, gamma, beta):
    '''
    Computes the functionality of batch normalisation with (in the given order)
    1) an affine transformation
    2) relu activation function
    3) batch normalisation
    The layer has

    Arguments:
    X - numpy array of shape (n1, batchsize)
    W - weights matrix of shape (?,?)
    b - bias vector
    gamma - scale parameters of shape (?,?)
    beta - shift parameters of shape (?,?)

    Returns:
    A - activations of layer for given mini-batch of shape (?,?)
    '''

    # Hyper Parameters
    # eps - to avoid division by zero, for numerical stability
    eps = 10^(-8)

    ### START CODE HERE ###




    ### END CODE HERE ###
    return A
```

Batch-Normalisation (2)

Assume we have a layer with n=3 units and relu activation function. Assume that we have a mini-batch with m=4 samples. The logit values are given by the matrix on the right.

$$\mathbf{z} = \begin{pmatrix} 12 & 14 & 14 & 12 \\ 0 & 10 & 10 & 0 \\ -5 & 5 & 5 & -5 \end{pmatrix}$$

- Compute the normalised z-values (z_{norm}) and the activations of the layer (expressed as a matrix) by assuming the scale and shift parameters introduced by batch norm to be 1 or 0, respectively (apply batch norm on z before applying 'relu').
- How many parameters (for scale and shift) are added by batch normalisation?
- Give two benefits of using a batch normalisation.
- Describe how batch-normalisation is handled at test or production time.

Regularisation

- Specify the mean-square error loss function with L2 regularisation (as a function of the activations of the last layer). How is gradient descent update rule modified by the regularisation?
- Why do we often refer to L2-regularization as “weight decay”? Derive a mathematical expression to explain your point.
- Explain the steps to be performed (at training and test time) when implementing dropout and explain why dropout in a neural network acts as a regulariser.
- Give a method to fight exploding gradient in fully-connected neural networks.

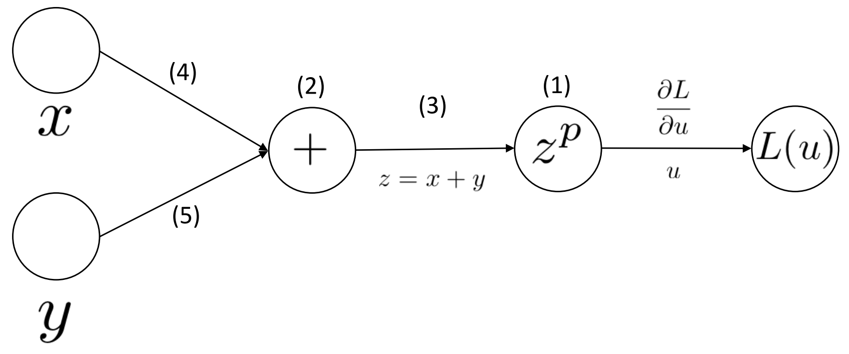
Comp Graphs, Backprop (1)

- Draw the computational graph for the function.

$$f(x, w, b) = 1 + \frac{1}{1 + (w \cdot x + b)^2}$$

At the nodes only the operations +,-,*,/ and powers $(.)^p$ are permitted.

- For the computational graph depicted below, specify the local gradients at the nodes (1), (2) and the gradients on the edges (3)-(5) - the latter by using $\frac{\partial L}{\partial u}$



- Compute the local gradient for a max pooling layer and given input into the pooling layer [to be further specified].

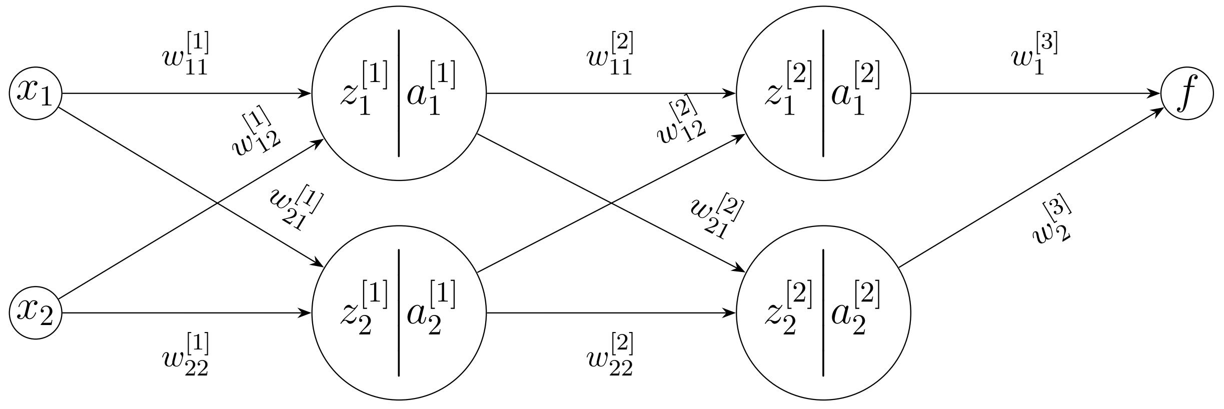
Comp Graphs, Backprop (2)

For the network with two hidden layers as depicted on the right and with

$$f = w_1^{[3]} a_1^{[2]} + w_2^{[3]} a_2^{[2]}$$

compute the derivatives

$$\frac{\partial f}{\partial z_1^{[2]}} \text{ and } \frac{\partial f}{\partial w_2^{[2]}}$$



$$Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad , \quad A^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \end{bmatrix}$$

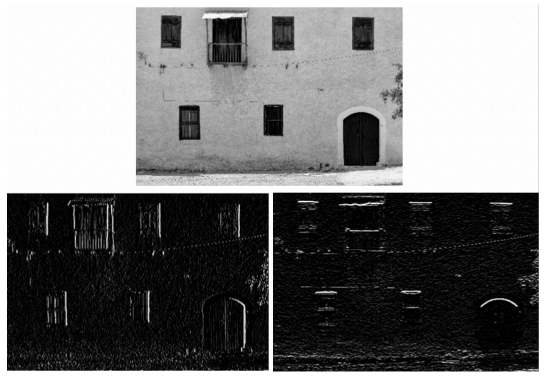
$$Z^{[2]} = \begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} \quad , \quad A^{[2]} = \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[2]}) \\ \sigma(z_2^{[2]}) \end{bmatrix}$$

CNNs (1)

- Consider a convolutional layer with k=10 filters of f=width=height=5, padding p=0 and stride s=2, 'relu' activation
  - Compute the shape of the output if the input has shape 28x28x3.
  - Compute the number of parameters that need to be trained.
- For input shape 28x28x3 compute the shape of the output for the following network:
  - k=10 filters with f=width=height=5, padding p=1 and stride s=2, 'relu' activation function
  - max pooling with stride s=2
  - k=15 filters with f=width=height=3, padding p=2 and stride s=2, 'relu' activation
  - 20 filters with width=height=1, padding p=0 and stride s=1, 'relu' activation
- What padding needs to be used for a convolutional layer with 3 filters of f=height=width=5 and stride s=1 applied to input images of shape 28x28x3 so that the output and input shape are the same?
- Compute the result of applying a filter followed by a max pooling. What information needs be cached so that it can be used during back-propagation?

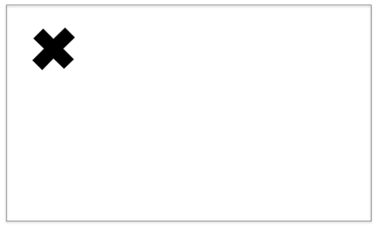
CNNs (2)

- Which of the following is true about max-pooling?
  - It allows a neuron in a network to have information about features in a larger part of the image, compared to a neuron at the same depth in a network without max pooling.
  - It increases the number of parameters when compared to a similar network without max pooling.
  - It increases the sensitivity of the network towards the position of features within an image.
- Look at the grayscale image at the top of the collection of images on the right.
  - Deduce what type of convolutional filter was used to get each of the lower images.
  - Explain briefly and include typical values of these filters. The filters have a shape of (3,3).
  -

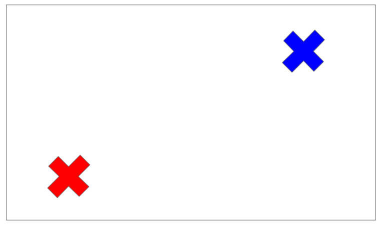


CNNs (3)

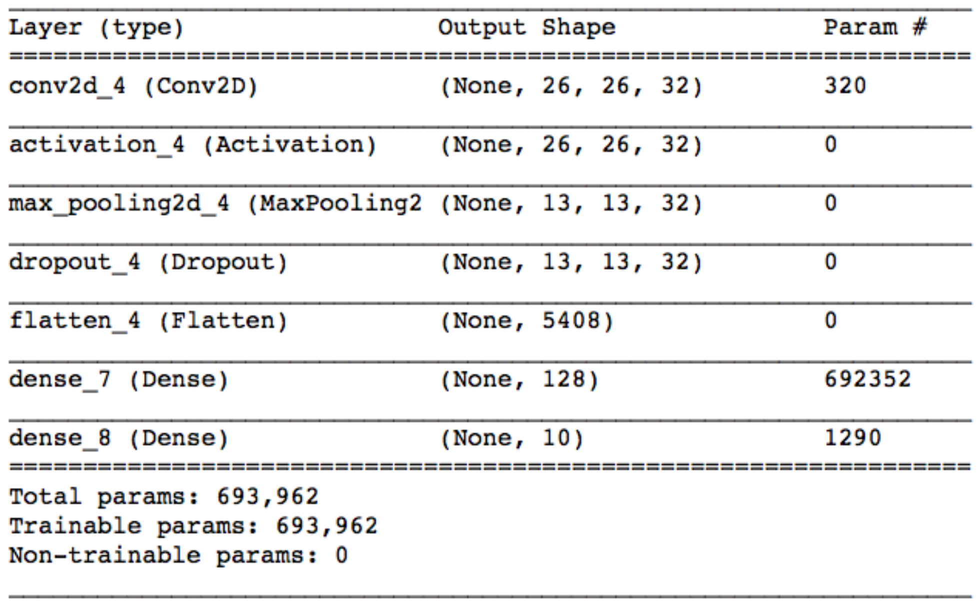
- Hand-engineer a filter to be convolved over the grey-level image (single channel) that leads to activations in the output at the position of the 3x3 black cross.



- Hand-engineer a filter to be convolved over the image (three channels, rgb) that leads to activations in the output at the position of the 3x3 red cross but not the blue cross.



- Assume that a model specified in `keras` produces the summary output as depicted on the right.
  - Provide the code (in `keras`) to build such a model.
  - Explain the role and configuration of each layer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
activation_4 (Activation)    (None, 26, 26, 32)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 5408)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               692352
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
_________________________________________________________________
```

CNNs (4)

- Describe as precisely as possible what the filters depicted below are designed to detect.:

a)

0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0

b)

0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

c)

1	1	1	1
1	0	0	0
1	0	0	0
0	0	0	0

- Specify a 3x3 filter good for detecting horizontal edges.

- Identify possible issues in a given `keras` code and describe how to fix them. (wrong loss function, shapes not matching, etc.)

- What is the idea behind using 1x1 convolution layers?
- Does a 1x1 convolution layer applied to an input layer with shape 1x28x28 (MNIST) make sense?

CNNs (5)

- In the context of visualising what is happening in deep network, explain how activation maximisation works and write down pseudo code for implementing it.

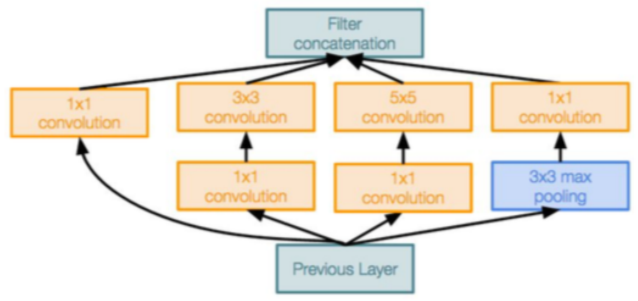
- List potential data augmentation strategies for classifying MNIST data. Sketch a plot with the learning curves illustrating what effect you would expect for the accuracy in comparison to a training without data augmentation..

- Consider a simple convolutional neural network with one convolutional layer. Which of the following statements is true about this network? (Check all that apply and provide explanations for the checked answers.)
  1. It is scale invariant (*).
  2. It is rotation invariant.
  3. It is translation invariant.
  4. All of the above.

(*) A model is said to be invariant under a transformation if the model applied to transformed input leads to similar output as when transforming the output from the model applied to untransformed input (up to within boundary effects).

Deep Network Architectures

- Explain two key innovations introduced by GoogLeNet?
- Describe the structure of the new type of layer introduced with GoogleNet and explain the ideas behind.
- Compute the number of (trainable) parameters involved when applying an inception module (as depicted on the right) to a layer with 10 filters. Furthermore, compute the output dimension of the layer by assuming that the input shape is 10x30x30.
- VGG and GoogleNet increased the depth significantly as compared with previous models. What is GoogleNet's solution to deal with the problems with backprop applied in deep architectures?



Inception module

Deep Network Architectures, Transfer Learning

- Describe the key ingredients of the ResNet architecture. What was the key innovation provided by ResNet? Describe the key elements that helped to train the ResNet architecture with 152 layers.

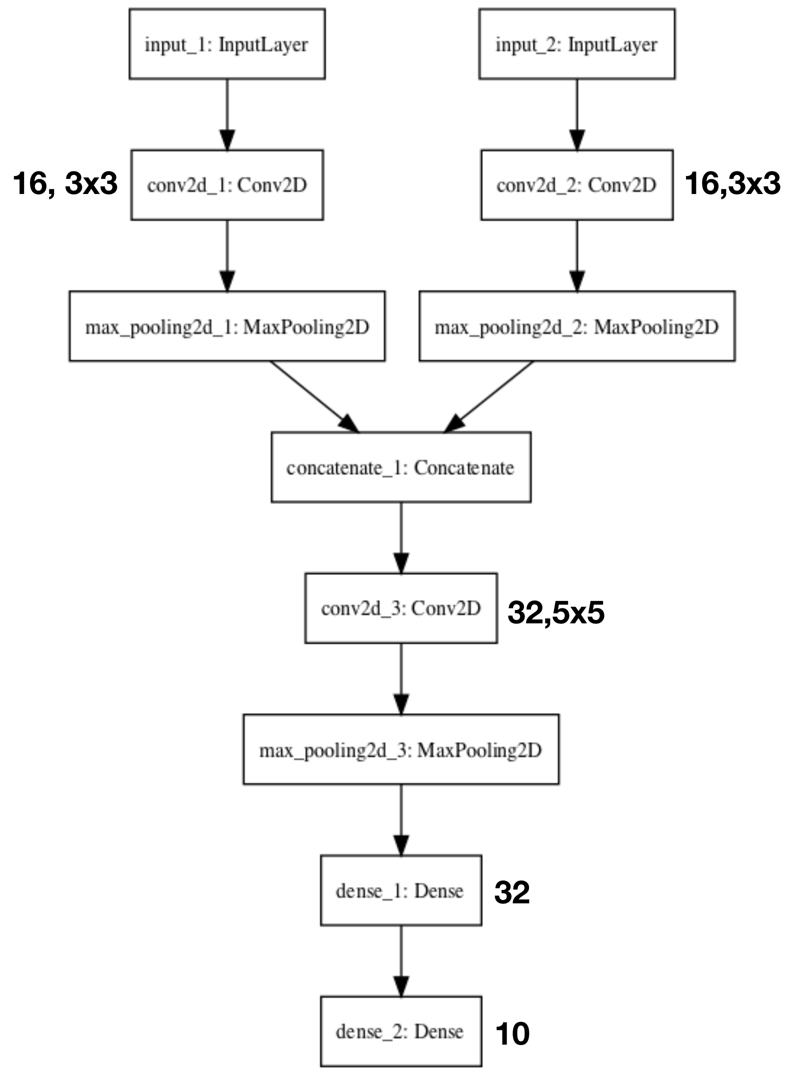
- Assume you should design a NN for classifying images of flowers. You have only limited labeled data of flowers available and the application should run on a mobile phone.
  - What basic architecture would you choose?
  - What data preprocessing steps typically are necessary?
  - Sketch a flow diagram with all the steps needed in the processing.

keras Functional API

- Explain what the following code is doing (assume you don't have the lines with comments). Create a plot with a graph that illustrates this.

```
visible = Input(shape=(28,28,1))
# first feature extractor
conv1 = Conv2D(32, kernel_size=3, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
flat1 = Flatten()(pool1)
# second feature extractor
conv2 = Conv2D(32, kernel_size=5, activation='relu')(visible)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat2 = Flatten()(pool2)
# merge feature extractors
merge = concatenate([flat1, flat2])
# interpretation layer
hidden1 = Dense(100, activation='relu')(merge)
# prediction output
output = Dense(10, activation='softmax')(hidden1)
model4 = Model(inputs=visible, outputs=output)
```

- Sketch the `keras` code that would implement the model as depicted in the graph on the right. The model takes pairs of images of shape 28x28x1 (as MNIST) and the number and shape of filters is indicated next to the boxes.
  Also compute the number of trainable parameters for this model.
  Bonus question: What purpose could this model have?



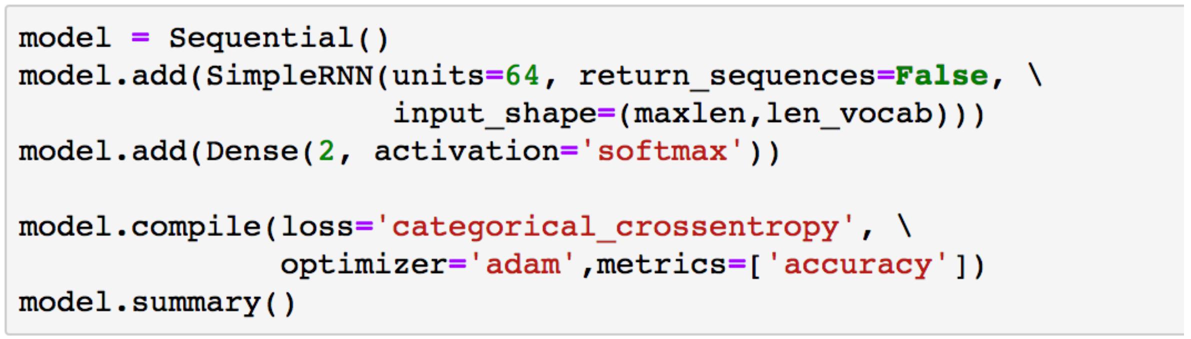
- Outline an image classification model that accepts pairs of images of as inputs (assuming two cameras positioned differently). The elements in the pairs are of different shape. Draw a suitable network graph and sketch how this could be implemented in `keras`.

Auto-Encoders

- Design a NN model for de-noising images.
  What would be the steps to train such a model?
- List typical applications for auto-encoders and describe why and how auto-encoders can be used in these applications.

Recurrent Nets

- List 5 typical applications for RNNs and explain why for these applications RNNs may help.
- In what sense are recurrent layers different from convolutional layers? How are parameters shared in RNNs and in CNNs?
- Assume you have designed a RNN for classifying sequences. What are typical data preparation steps?
- Calculate the number of trainable parameters for the model specified with the given code snippet (in `keras`).



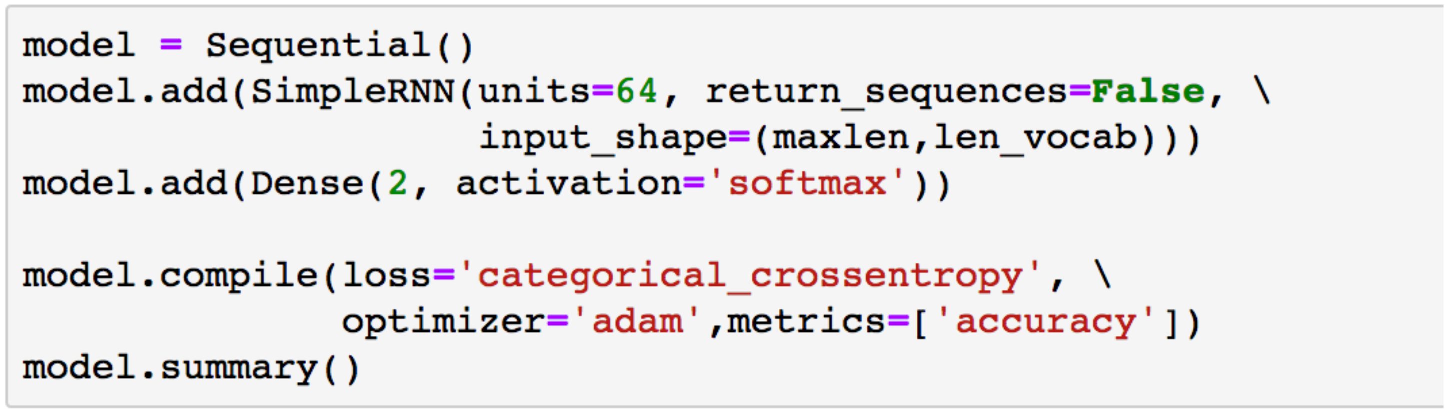
```
model = Sequential()
model.add(SimpleRNN(units=64, return_sequences=False, \
                    input_shape=(maxlen,len_vocab)))
model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical_crossentropy', \
              optimizer='adam',metrics=['accuracy'])
model.summary()
```

- Describe the problems typically encountered when training SimpleRNNs and how these can be encountered / solved.
- Explain with math formulas how back-propagation affects the calculation of the gradient w.r.t. the weights matrices of a Simple RNN layer.

Recurrent Nets

- Describe what is understood as the problem of long-term dependencies, explain why it occurs and describe how LSTMs and GRUs help to reduce it. Are there other techniques that may also help?
- Add regularisation to the model specified with the given code snippets. Explain precisely what you are doing.



```
model = Sequential()
model.add(SimpleRNN(units=64, return_sequences=False, \
                    input_shape=(maxlen,len_vocab)))
model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical_crossentropy', \
              optimizer='adam',metrics=['accuracy'])
model.summary()
```

Case Study 1

DeepMind recently invented AlphaGo Zero – a Go player that learns purely through self-play and without any human expert knowledge. AlphaGo Zero was able to impressively defeat their previous player AlphaGo, which was trained on massive amounts of human expert games. AlphaGo in its turn had beat the human world Go champion- a task conceived nearly impossible 2 years ago! Unsurprisingly, at its core, AlphaGo Zero uses a neural network.

Your task is to build a neural network that we can use as a part of AlphaTicTac-Toe Zero. The board game of TicTacToe uses a grid of size 3×3, and players take turns to mark an $\times$ (or $\bigcirc$) at any unoccupied square in the grid until either player has 3 in a row or all nine squares are filled.

(a) (2 points) The neural network we need takes the grid at any point in the game as the input. Describe how you can convert the TicTacToe grid below into an input for a neural network.

×	○	
	×	×
	○	

(b) (3 points) The neural network we require has 2 outputs. The first is a vector $\vec{\boldsymbol{a}}$ of 9 elements, where each element corresponds to one of the nine squares on the grid. The element with the highest value corresponds to the square which the current player should play next. The second output is a single scalar value $\boldsymbol{v}$, which is a continuous value in [-1,1]. A value closer to 1 indicates that the current state is favorable for the current player, and -1 indicates otherwise.

Roughly sketch a fully-connected single hidden layer neural network (hidden layer of size 3) that takes the grid as input (in its converted form using the scheme described in part (a)) and outputs $\vec{\boldsymbol{a}}$ and $\boldsymbol{v}$. In your sketch, clearly mark the input layer, hidden layer and the outputs. You need not draw all the edges between two layers, but make sure to draw all the nodes. Remember, the same neural network must output both $\vec{\boldsymbol{a}}$ and $\boldsymbol{v}$.

(c) (i) (1 point) As described above, each element in the output $\vec{\boldsymbol{a}}$ corresponds to a square on the grid. More formally, $\vec{\boldsymbol{a}}$ defines a probability distribution over the 9 possible moves, with higher probability assigned to the better move. What activation function should be used to obtain $\vec{\boldsymbol{a}}$?

(ii) (1 point) The output $\boldsymbol{v}$ is a single scalar with value in [-1,1]. What activation function should be used to obtain $\boldsymbol{v}$?

(d) (4 points) During the training, given a state t of the game (grid), the model predicts $\vec{\boldsymbol{a}}^{<t>}$ (vector of probabilities) and $\boldsymbol{v}^{<t>}$. Assume that for every state t of the game, someone has given you the best move $\vec{\boldsymbol{y_a}}^{<t>}$ to do (one-hot vector) and the corresponding value $\boldsymbol{y_v}^{<t>}$ for $\boldsymbol{v}^{<t>}$

In terms of $\vec{\boldsymbol{a}}^{<t>}$, $\boldsymbol{v}^{<t>}$, $\vec{\boldsymbol{y_a}}^{<t>}$ and $\boldsymbol{y_v}^{<t>}$, propose a valid loss function for training on a single game with T steps. Explain your choice.

Case study 2 - bird recognition

- You are approached by the swiss national ornithology association: SNOA. They would like to build a system to identify birds in natural settings.
  - They want to use 3 cameras positioned each on three different trees pointing to a scene on the ground. The cameras are triggered at the same time by a movement sensor, i.e. you have for each detection 3 pictures of the bird with different angles.
  - The SNOA wants to identify 52 different bird types for which they provided the labels on the dataset. Images may contain other types of birds (out of the 52, labelled as UNKNOWN) or contain nothing if the movement sensor triggers unexpectedly.
  - The dataset contains approximately 1500 images per bird type.
- Explain your strategy for building a bird recognition system.
  - Would you use deep learning, if yes how?
  - What type of architecture would you use?
  - How would you prepare the data?
  - How would you evaluate the system?