Miscellaneous Material of

Artificial Neural Networks and Deep Learning

From Professor Giacomo Boracchi and Matteo Matteucci's lectures
for the MSc in Mathematical Engineering

by Teo Bucci

Politecnico di Milano
A.Y. 2022/2023

Document created on January 7, 2023

Developed by:
Teo Bucci - `teo.bucci@mail.polimi.it`

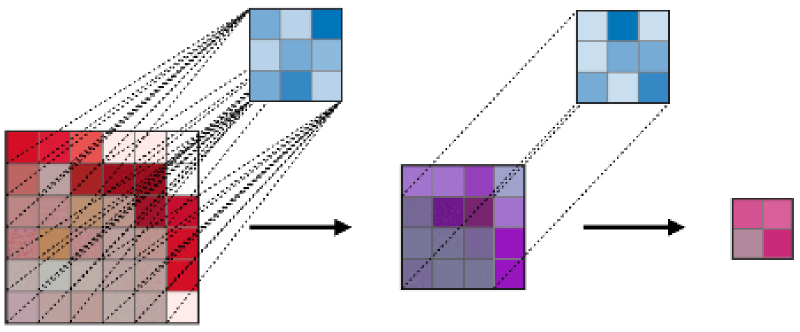
Compiled with ♡

Contents

Receptive field. The receptive field at layer k is the area denoted $R_k \times R_k$ of the input that each pixel of the k-th activation map can 'see'. By calling F_j the filter size of layer j and S_i the stride value of layer i and with the convention $S_0 = 1$, the receptive field at layer k can be computed with the formula:

$$R_k = 1 + \sum_{j=1}^{k} (F_j - 1) \prod_{i=0}^{j-1} S_i$$

In the example below, we have $F_1 = F_2 = 3$ and $S_1 = S_2 = 1$, which gives $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$.



1 Syllabus

The following is a list of topics, questions, subjects which are expected you to know at the end of the course on Artificial Neural Networks and Deep Learning. For this reason you can use it to double check your preparation before the exam. Please recall the exam is meant to verify what you have understood from the slides and lecture provided by the teachers, having said this, note:

- This is NOT the list of questions and exercises you will find in the exam, but we will look at this list when preparing it, so the exam questions are going to check these points
- If you know how to answer/face any of the following then you know how to face any possible question for the exam

Machine vs Deep Learning

- What is machine learning
- What are the machine learning paradigms and their characteristics
- What are the machine learning task and their characteristics
- What is deep learning and how it differs from machine learning
- Examples of successful deep learning applications
- What is transfer learning and what it is used for
- What is a feature and its role in machine and deep learning
- What is cross-validation and what it is used for
- What are hyperparameters and identify those for all the models presented in the course
- What is Maximum likelihood estimation and how does it work in practice
- What is Maximum a posteriori and how does it work in practice

Feed Forward Neural Networks

- The perceptron and its math
- The Hebbian Learning paradigm, its maths, and it application
- The feed forward neural architecture and its improvement wrt the Perceptron
- The number of parameters of any model you can build
- Activation functions, their math, their characteristics, and their practical use

- What is backpropagation and how does it work
- The relationship with non-linear optimization and gradient descent
- Backpropagation computation for standard feedforward models
- Online vs batch, vs mini-batch learning and their characteristics
- Forward-backward algorithm for backpropagation
- Derivatives chain rules
- Error functions, their statistical rationale, their derivation, and their derivative
- The issue of overfitting and its relationship with model complexity
- Regularization techniques, their rationale and their implementation
- Techniques for hyperparameters tuning and model selections
- Techniques for weights initialization and their rationale
- Batch-Normalization rationale and use

Convolutional Neural Networks

- Convolution and correlation, linear classifier for images
- The layers of a CNN
- Connections between CNN and Feedforward-NN, interpretations of CNN
- How to compute the number of parameters of a CNN
- Best practices to train CNN models: data augmentation, Transfer learning
- Design criteria from the most successful architectures shown during classes (no need to know exactly the architectures)
- Fully convolutional CNN and CNN for segmentation
- The key principles of CNN used for object detection
- Residual learning
- GANs, what they are and how they are trained Recurrent Neural Networks
- Models for sequence modeling
- The architecture and math of a recurrent neural network
- Backpropagation through time rationale and math
- The limits of backpropagation through time and the vanishing/exploding gradient issue
- The vanishing gradient math in a simple case
- The Long Short-Term Memory model, rationale, ad math
- The Gated Recurrent Unit
- LSTM Networks, hierarchical and bidirectional.

Sequence to Sequence Learning

- Sequential Data Problems, with examples
- The Seq2Seq model, training, and inference
- Neural Turing Machine model and the attention mechanism
- Attention mechanism in Seq2Seq models
- Chatbot: core models and context handling
- The Transformer model

Word Embedding

- Neural Autoencoder model, error function, and training
- Language models and the N-grams model

- Limits of the N-gram model
- The concept of embedding and its benefits
- The Neural Language Model and its use for word embedding
- Google's word2vec model (CBOW)
- Possible uses of word embedding

2 Introduction

2.1. Does learning rate have some relationship with the dying neurons issue? Why? (0.5/1 Pts.)

No. It doesn't. Because the dying neuron problem is due to the ReLU activation function and happens when a neuron is stuck in the negative part of the domain, where it cannot learn anything. In this case, modifying the learning help won't be of any help, because the problem lies in the ReLU activation function and we need to change that to Leaky ReLU in order to fix this issue.

2.2. Which of the following is true (0.5/1 Pts.)

A low learning rate does not compromise training
✓ A high learning rate might compromise training
The higher the learning rate the better the training
✓ The learning rate should be reduced while training

2.3. One of your friends shows up with a brand new model she has invented, but which she is not able to train. You suspect this could be due to VANISHING/EXPLODING GRADIENT, what would you look at to check THIS hypothesis? HINT: Six and only six are correct; if you mark more, these will be counted as errors, if you mark less you are loosing points. (3 Pts.)

✓ The activation functions used for neurons
The output values of neurons during training
✓ The distribution of the weights during training
✓ The initialization of the weights
✓ The depth of the network
The balance of classes
The gradient descent algorithm used
✓ The average gradient norm during training
The value of the learning rate
Whether the model contains Convolutional layers
✓ Whether the model contains Recurrent Layers
The loss function used

2.4. Which of these techniques might help with overfitting? (1/1 Pts.)

✓ Early Stopping
✓ Weight Decay
✓ Dropout
Xavier Initialization

✓ Batch Normalization
ReLu and Leaky ReLu
Stochastic Gradient Descend

2.5. Briefly describe what Siamese networks are and what they are used for. (2/2 Pts.)

Siamese networks is a concept that arise in the field of Metric Learning. In particular, they can be found in tasks of template matching. In these tasks, we want to compare a template with an image to verify if it is one of the templates we have or not. E.g. suppose that we want to open a door automatically only if an employee comes to work, we will have an image of the employ and a camera taking a real image of the person in front of the door. Since the data are small, we cannot learn a network performing the task directly because we have few samples and we should retrain the model at each employee change. Then, Siamese network solve the problem. We train a CNN on a large dataset to recognize the elements we want to recognize to be able to produce an embedding. When the network is finished, we remove the fully connected network on top of it and we copy the network: we have two identical copies of the same network performing the embedding. At test time, we generate the embedding of the sample and of the templates with the two networks: one embed templates and one the real sample. Then, we will classify the sample as the template to which the embedding is closest.

2.6. What is the dying neuron problem and how would you fix it? (2 Pts.)

The dying neuron problem is a problem we have with ReLU activation function. Since the ReLU activation function is defined as f(x)=max(Ox), it is null in case $x < 0$. Its gradient is always unitary when f(x)=x and it becomes null for <0. If the updates make a neuron go in that part of the space (x<0), the neuron will die by being turned off. This is called dying neuron problem. We can fix the dying neuron problem by slightly changing the ReLU activation function. The ReLU activation function can be redefined to be f(x)=0.001x for x<0 and f(x) =x for x> =0. In this case, the gradient will never be null and the problem of the dying neuron is solved since the neuron will only enter a region in which it is lower. However, there still remains the problem of non differentiability in O. Then, we can use a modification of the ReLU activation function called ELU. It solves both the vanishing gradient problem and the non differentiability problem of the ReLU function in 0.

2.7. Which of these techniques might help with vanishing gradient? (1/1 Pts.)

Early Stopping
Weight Decay
Dropout
✓ Xavier Initialization
✓ Batch Normalization
✓ ReLu and Leaky ReLu
Stochastic Gradient Descend

2.8. Make an example of an application where Classical UNSUPERVISED learning is used and than present its Deep counterpart. -> (I want it a real, short. fully specified, application example, including the algorithms and the models ... there should not be another answer like your in the class). (2 Pts.)

2.9. Which conceptual difference does make Deep Learning differ significantly from being just another paradigm of Machine Learning similarly to supervised learning, unsupervised learning, reinforcement learning, etc.? (1 Pts.)

2.10. Make an example of an application where Classical SUPERVISED learning is used and than present its Deep counterpart. -> (I want it a real, short. fully specified, application example, including the algorithms and the models . . . there should not be another answer like your in the class). (2 Pts.)

2.11. Make an example of an application where Classical UNSUPERVISED learning is used and than present its Deep counterpart. -> (I want it a real, short. fully specified, application example, including the algorithms and the models ... there should not be another answer like your in the class). (2 Pts.)

2.12. What are the major differences when training feed forward neural networks for regression and for classification? Motivate your answer (20 Pts.)

2.13. Let's start from the core issue of machine learning, i.e, the overfitting/generalization trade-off. Discuss if and how deep learning (as a general paradigm) differs from (classical) machine learning regarding the overfitting issue. Hints: start by defining the concepts, then highlight the differences, then connect if/how these differences are related to overfitting/generalization. (30 Pts.)

The issue of overfitting happens when a machine learning model or a dep learning network is being trained. This issue happens when the model is learning the training data too much and in doing it loses the ability to generalize. This issue happens in machine learning when the model is too complex and the model starts to model not only the structure behind the data but also the noise over it. A deep neural network can compute any function with the right weights and the right number of neurons and deep enough models, it usually happens that the network is a really complex model and the risk of overfitting is high. The issue for neural network is also that it can end up learning features which are too specific for the training set and which does not generalize. In traditional machine learning this issue is more connected with the complexity of the model which uses the hand crafted features

2.14. Which loss function should be used in regression according to the Maximum A-Posterior approach? Why?

2.15. Which loss function should be used in regression according to the Maximum Likelihood approach? Why? (10/15 Pts.)

Grazie alla MLE abbiamo capito che la migliore loss function per la regressione è la mean squared error. Questo deriva della probabilità condizionata degli input dati i pesi, di cui viene tatto il logaritmo e successivamente calcolato il gradiente rispetto ai pesi e posto uguale a zero.

According to MLE the best function for regression is sum of squared errors. This is due to the assumption the output target t follows a Gaussian distribution ast g(x|w) + N(O,sigma). Given N lid samples from t and computing the likelihood of the sample we derive that the maximum is obtained in case of the minimization of the sum or squared errors.

2.16. What are the issues of the ReLU activation function? How could you solve them? (15/20 Pts.)

I principale problema della relu è che se dovessimo trovarci nella parte «O della funzione l'output sarebbe pari a zero e rischieremmo di avere un vanishing gradient. Per ovviare a questo problema si può: Usare una leaky ReLu, che non soffre di questo problema Usare un batch gradient descend, che diminuicro la nrahabilità di trovarci nella parte. . .

2.17. How would you decide for the activation function of the output layer of a deep neural network? (20/20 Pts.)

Dipende dalla task che la rete deve svolgere, se abbiamo un classificatore binario le più adatte sono sigmoide e tangente iperbolica, se abbiamo un classificatore generico (n classi) la migliore è la softmax, se invece abbiamo un regressore la migliore è una funzione lineare o una relu (nel caso in cui il valore non potesse essere inferiore a zero.

2.18. Make an example of deep learning model, i.e., a deep neural network, for unsupervised learning describing it briefly in terms of goal, architecture and loss function.

2.19. Make an example of deep learning model, i.e., a deep neural network, for supervised learning describing it briefly in terms of goal, architecture and loss function. (20/20 Pts.)

Un esempio di modello di DL con supervised learning è un modello usato per la classificazione di immagini. Infatti nel training set il modello riceve delle immagini con assegnate delle etichette prese da un insieme prestabilito (rispettivamente input e target output). Lobiettivo del modello è imparare ad associare a ogni immagine l'etichetta (tra le etichette del nostro insieme) che la rappresenta. Le architetture possono essere diverse, se usiamo un classificatore lineare l'architettura sarà un feed forwrd neural network, ma possiamo usare anche delle convolutional neural networks, avendo anche dei layer convoluzionali e di pooling nell'architettura. In entrambi i casi la loss function sarà una crossentropy, la più adatta per la classificazione.

2.20. What is the difference between supervised learning and unsupervised learning? (5/10 Pts.)

Nel supervised learning il modello riceve una serie di dati e di target input e impara, dato un nuovo dato fuori dal training set, a dare un output abbastanza buono, mentre nel caso dell'unsupervised learning riceve solo dei dati e sfrutta le caratteristiche comuni nei dati per ...

2.21. What is the difference between machine learning and deep learning? (10/10 Pts.)

(10/10 Pts.)

La sostanziale differenza tra ML e DL è che il primo ha bisogno di un feature engineer per estrarre delle feature, mentre il secondo lo fa automaticamente in modo data-driven. Più nello specifico, nel caso del ML un feature engineer scova, grazie a conoscenze pregresse e dati esterni, dele feature che serviranno al modello per risolvere un task, mentre nel DL il modello stesso cerca queste feature nei dati usati per 'allenamento, estraendo feature sempre più astratte, addirittura ininterpretabili per l'occhio umano.

2.22. Many architectural details are involved in obtaining a successful neural network model. For each of the following, list and discuss briefly the available options:

- **Output activation function**
- **Loss function**
- **Weight initialization**
- **Regularization**

2.23. With reference to Feed Forward Neural Networks answer the following questions:

- **What is the difference between supervised and unsupervised learning?**
- **Make an example of neural network used for supervised learning, i.e., describe the problem faced, the model, and the loss function used**
- **Make an example of neural network used for unsupervised learning, i.e., describe the problem faced, the model and the loss function used**

2.24. With reference to a Feed Forward Neural Network with 3 input, 1 hidden layer having 5 hidden neurons, and an output layer with 2 neurons.

- **Draw the network, and provide its output characteristics, i.e., the mathematical formula, for the output of the previous network as a function of its input and weights**
- **Consider the previous network to be used for classification. Define its activation functions and error/loss function providing motivations for your choices**
- **What is backpropagation and how does it work?**

2.25. Which of the following statements about instance segmentation and semantic segmentation is true? (1.5 Pts.)

Instance segmentation and semantic segmentation are different problems solved by the same networks
✓ In order to train an instance segmentation network, it is necessary to fine tune a semantic segmentation network first
?? Instance segmentation network returns both segments and bounding boxes
✓ Architectures for semantic segmentation can have a shape similar to a convolutional autoencoder
Fully convolutionalization is a way to modify without training a CNN classifier, to become a (very coarse) instance segmentation network

2.26. Which of the following statements about Global Averaging Pooling is true (1.5 Pts.)

✓ GAP has no trainable parameters
GAP should not be used in networks containing a batch normalization layer
GAP is used in object detection networks
Networks provided with GAP need to be trained to return class activation mapping
✓ Including a GAP in a CNN is a good way to make the CNN invariant to input size

2.27. What is Xavier initialization and why it helps with vanishing gradient? (15 Pts.)

2.28. Backpropagation suffers the vanishing problem issue. What it is and how to solve it? (20 Pts.)

2.29. Backpropagation suffers the exploding gradient issue. What it is and how to solve it? (20 Pts.)

2.30. Which loss function should be used in classification according to the Maximum Likelihood Estimation? Why? (10 Pts.)

2.31. What is Transfer Learning and what it is used for? (Be short and focused, but answer both!) (20 Pts.)

2.32. What is the Global Average Pooling? Where would you introduce GAP in the above network and how would you modify the network accordingly? List the modified network architecture, one layer per row. (30 Pts.)

2.33. When would you prefer weight decay with respect to early stopping? How can you tune the gamma parameter of weight decay? (2 Pts.)

2.34. What is it and how can you tune the gamma parameter of weight decay? (15/20 Pts.)

The gamma parameter in weight decay balances the regularization effect of the L2 term with respect to the fitting term. It can be tuned by cross-validation, i.e., evaluating different gamma values with a holdout set and then training with the best gamma using all the data.

3 CNN

Consider the following Python snippet

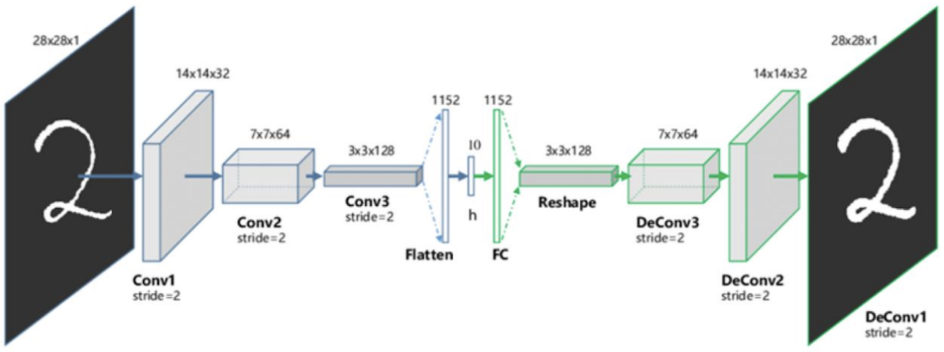
```python
import keras
from keras.utils import np_utils
from keras.models import Sequential, Model
from keras.layers import Dense, Activation, Flatten, Conv2D,
from keras.layers import MaxPooling2D, BatchNormalization
pool_size = (2,2) # size of pooling area for max pooling
nb_filters = 64

model = Sequential()

model.add(Conv2D(nb_filters, (11,11), input_shape=(64, 64, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Conv2D(nb_filters*2, (5,5), padding='same')) # 2nd Conv Layer starts
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Conv2D(nb_filters*4, (3,3), padding='same')) # 3rd Conv Layer starts
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Conv2D(nb_filters, (1,1), padding='same'))   # 4th Conv Layer starts
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = pool_size))
model.add(Flatten())
model.add(Dense(64))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(21))
model.add(Activation('softmax'))
model.summary()
```

3.35. Answer the following questions providing a short explanation for each

- **Is this a classification or a regression network? How many output?**
- **The following is the output of the `model.summary()` command. A few numbers have been replaced by dots. Please, fill all the numbers in and in particular indicate which computation you are performing to get to each number. Note: the first dimension is the batch size, and as such this is set to None.**
- **How big is the receptive field of the pixel at the center of the feature map after the first convolutional layer? How about the pixel at the center of the second convolutional layer?**
- **This network has relatively large filters in the first layers. Can we replace that layer with others to reduce the number of parameters and at the same time preserve (or increase) the receptive field?**

3.36. Enumerate the building blocks of the networks you would design to build the Denoising Autoencoder as if you were going to implement it in Keras (no need to write the Keras code) and for each of them tell the number of parameters providing a short description on how you computed them, e.g., 3x5x5=45 and not just 45. (Yes you can use the calculator, but we are more interested in the formula than on the numbers) (30 Pts.)

A simple architecture would be: 6 blocks of convolution with 3 by 3 kernels and depth starting at 8 and doubling each block, followed by max pooling these would have (3x3x1x8+8)+(3x3x8x16+16)+(3x3x16x32+32)+(3x3x32x64+64)+(3x3x64x128+128)+(3x3x128x256) parameters. At this point we have a 4x4x256 tensor which represents the image, which we can flatten and pass through a first dense layer with 1024 neurons and a second one with 256 neurons, before finally arriving at the compressed representation with a third dense layer with 8 neurons. This part would have (4096*1024+1024)+(1024x256+256)+(256x8+8) parameters. In the expanding branch of the network I would use the same logic and sizes, but I would ...



3.37. Detail the building blocks in each layer of the above network. Please indicate

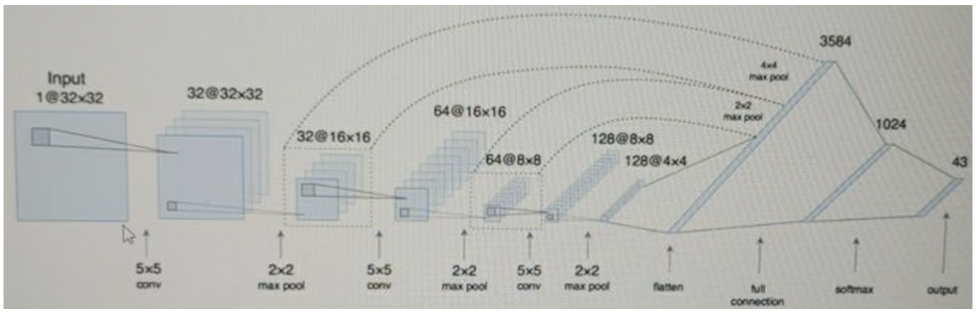
- **What kind of block is**
- **What is its size**
- **what is the overall number of parameters, together with a short description**

on how you compute them, e.g., 3x5x5=45 (not just 45!). Consider convolutions having a spatial 3x3 extent. You can use the calculator, but we are more interested in the formulas than on numbers! (20 Pts.)

3.38. This network is called denoising autoencoder and it has the goal of cleaning or restoring a damaged image. How would you train it? What kind of loss function would you use? (20 Pts.)

3.39. What is the relationship we learn in a neural autoencoder? Why we do it? (1 Pts.)

3.40. How could we size the embedding of a neural autoencoder? (1 Pts.)



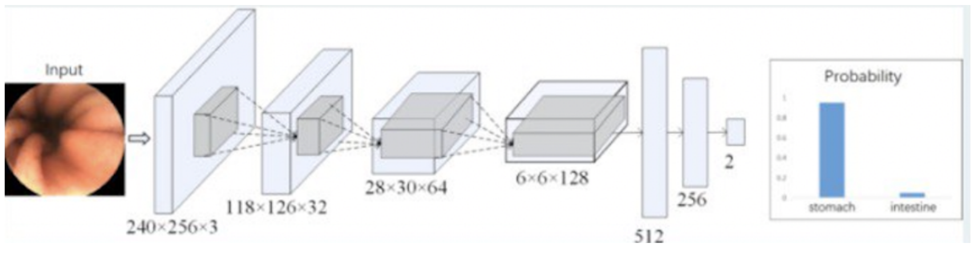
3.41. Enumerate the building blocks of the networks as if you were going to implement it in Keras and for each of them tell the number of parameters providing a short description on how you compute them, e.g., 3x5x5=45 and not just 45. (Yes you can use the calculator, but we are more interested in the formula than on the numbers !)

3.42. Can transfer learning help with the vanishing gradient issue? Why? (10 Pts.)

3.43. Are autoencoders unsupervised learning models? Answer by defining what unsupervised learning is and motivate your answer accordingly. (20 Pts.)

3.44. Describe a neural autoencoder and what it can be used for. (20 Pts.)

3.45. Can autoencoder overfit? Motivate your answer and, in case they do, explain how to detect it. (20 Pts.)



3.46. Enumerate the building blocks of the above network as if you were going to implement it in Keras. For each of them tell the number of parameters providing the formula to

compute them, e.g., 3x5x5=45 and not just 45. Please report any assumption you make for those information that are not provided in the picture. List each layer of the network in a different row of your answer to improve readability. (Yes you can use the calculator, but we are more interested in the formula than on the numbers!) (40 Pts.)

The network in the picture can be summarized as follows:

```
Input              # 240 * 256 * 3
Conv2D(32,5)       # output_shape: (None, 236, 252,32)  # p = 5 * 5 * 3 * 32 + 32
Maxpooling2D(2,2)  # output_shape: (None, 118, 126, 32) # p = 0
Conv2D(64,5)       # output_shape: (None, 114, 122, 64) # p = 5 * 5 * 32 * 64 + 64
Maxpooling2D(4,4)  # output_shape: (None, 28, 30, 64)   # p = 0
                                        # Actually 114/4 = 28.5 while 122/4 = 30.51
Conv2D(128,5)      # output_shape: (None, 24, 26, 128)  # p = 3 * 3 * 64 * 128 + 178
Maxpooling2D(4,4)  # output_shape: (None, 6, 6, 128)
                                        # Actually 26/4 = 6.51
Flatten            # output_shape: (None, 4608)
Dense              # output_shape: (None, 512)          # p = 4608 * 512 + 512
Dense              # output_shape: (None, 256)          # p = 512 * 256 + 256
Dense              # output_shape: (None, 2)            # p = 256 * 2 + 2
```

Assuming 3x3 filter, input to be GB and valid padding for convolutions. The dimensions are not always downscaled perfectly so I suppose a valid pad has been used. Like in the first block there is a /2 and then -2 so I suppose there is a stride = 2 3x3 conv that "leaves out" the borders and halves dimensions. In second block dimensions are down for like 4 times, so I suppose a stride =4 5x5 with always valid pad. Also in third block dimensions are downscaled so i suppose to be similar to previous block Conv2D(3x3, activation=relu, stride=2) params: (3x3x3 filter + 1 bias)x32 filters = 896 Conv2D(3x3, activation =relu, stride=4) params: (5x5x32 filter + 1 bias)x64 filters = 51264 Conv2D(3x3, activation=relu, stride=4) params: (5x5x64 filter + 1 bias)x128 filters = 204928 Flatten0 no params (output is 4608 vector) Dense(512) params: 4608x512 = 2359296 Dense(256) params: 512x256 = 131072 Dense(2) params: 256x2 = 512 Activation(softmax) no params (softmax to have [0, 11 scores)

3.47. Illustrate the major differences between a classification and an object detection network, and what are the major advantages of these latter over baselines built upon a CNN for classification. Consider R-CNN as a reference for object detection network. Describe what are the major changes introduced by Fast R-CNN and Faster R-CNN with respect to the baseline R-CNN (30 Pts.)

3.48. How data representation is learned via convolutional neural networks? (20 Pts.)

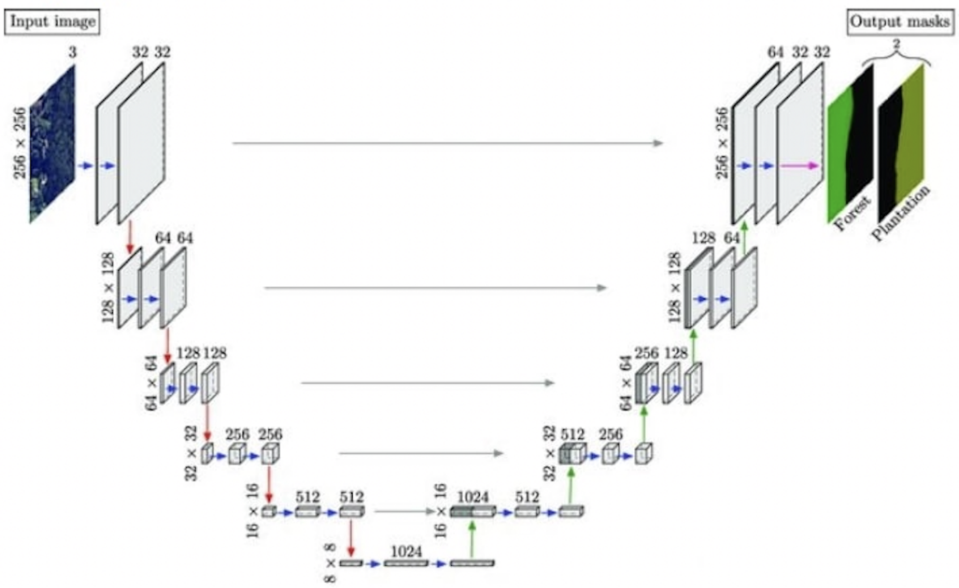
3.49. Which of the following transformations are expected to affect less the output of a CNN like the one depicted the figure above?

- **rotation**
- **translation**
- **intensity scaling**
- **zoom**

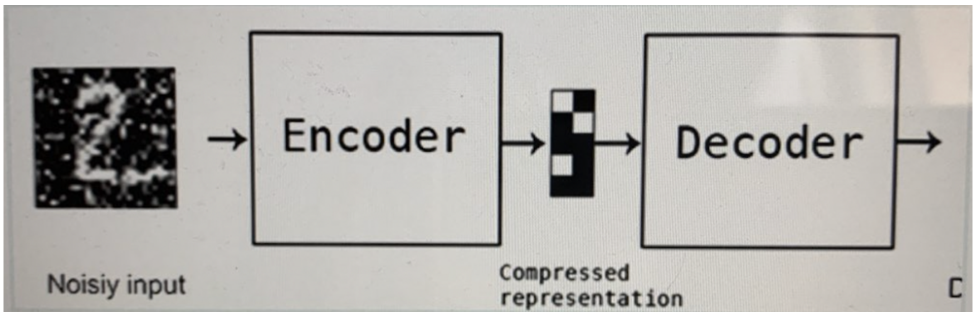
How can you improve invariance to those transformations that heavily influence the CNN output? PS: The question is not related to the type of images being fed to the network, but rather to the CNN architecture itself (20 Pts.)

Probably, insensity scaling and not so relevant translation/zoom transformation are less likely to influence the output, since they do not really create new features to analyze. Rotation and good translation can instead give the CNN example of different input, which actually allows the CNN to become more robust to different images in input - and therefore to generalise better. In fact, changing the orientation of the image actually gives CNN an hint about the fact that feature are not statically present in the image, but instead are properties that can be "shifted" in the net, and be present in different areas. Therefore, the learning of CNN is not static with respect to a precise zone of the image. To learn such feature in advance, it is possible with data auqumentation to provide the net with example of these type of transformed image. Augmentation allows to apply transformation to the image without changing the label, thus increasing the dataset size (which can be useful in case of data scarcety)

The transformations that affect less the output are rotation and translation, because the convolution operation is roto-translation invariant. This because it is the weighted sum of the weights of the filter and the portion of the input. To improve invariance it is possible to work on data, performing data augmentation on the training images, or by stacking several convolutional layers of the same size and depth, allowing to extract more-relevant features; for example, it is possible to stack three convolutional layers, with 32 filters each, before the Pooling layer. Data augmentation means modify the input with random transformations that allows the network to extract the features based on its content, rather than its position or rotation.



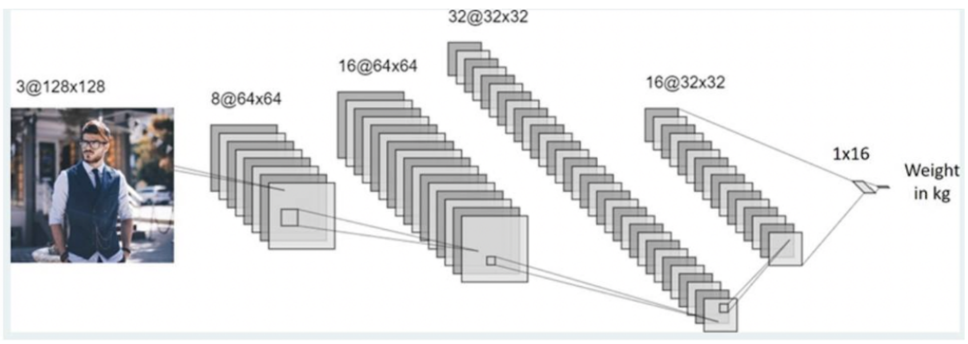
3.50. Enumerate the building blocks of this networks as if you were going to implement it, and for each block report the overall number of parameters together with a short description on how you compute them, e.g., 3x5x5=45 (not just 45!). Consider that blue arrows refer to convolutions having a spatial extent 3x3. while the magenta one does not perform spatial averages You can use the calculator. but we are more interested in the formulas than on numbers! (Note: for some of the blocks you might need to infer the number of channels or the size from the preceding/following blocks) (40 Pts.)



3.51. **In the picture, the general schema of a Denoising Autoencoder is reported with 256x256 greyscale images as input and a desired compressed representation of 8 float numbers. Enumerate the building blocks of the networks you would design to build the Denoising Autoencoder as if you were going to implement it in Keras (no need to write the Keras code) and for each of them tell the number of parameters providing a short description on how you computed them, e.g., 3x5x5=45 and not just 45. (Yes you can use the calculator, but we are more interested in the formula than on the numbers) (30 Pts.)**

```
x = Conv2D(128, (3,3), padding='same')(inp)     # p = 3 x 3 x 1 x 128 + 128 = 1280
                                                # out = 256 x 256 x 128
x = MaxPooling2D((2,2), padding='same')(x)      # p = 0
                                                # out = 128 x 128 x 128
x = Conv2D(64, (3,3), padding='same')(x)        # p = 3 x 3 x 128 x 64 + 64 = 73792
                                                # out = 128 x 128 x 64
x = MaxPooling2D((2,2), padding='same')(x)      # p = 0
                                                # out = 64 x 64 x 64
x = Conv2D(32, (3,3), padding='same')(x)        # p = 3 x 3 x 64 x 32 + 32 = 18464
                                                # out = 64 x 64 x 32
x = UpSampling2D(2,2))(x)                       # p = 0
                                                # out = 128 x 128 x 32
x = Conv2D(64, (3,3), padding='same')(x)        # p = 64 x 3 x 3 x 32 + 64 = 18496
                                                # out = 128 x 128 x 64
x = UpSampling2D(2,2))(x)                       # p = 0
                                                # out = 256 x 256 x 64
x = Conv2D(128, (3, 3), padding='same')(x)      # p = 128 x 64 x 3 x 3 + 128 = 73.856
                                                # out = 256 x 256 x 128
out = Conv2D(1, (3, 3), padding='same')(x)      # p = 3 x 3 x 1 x 128 + 1 = 1153
                                                # out = 256 x 256 x 1
```

3.52. **Describe the training procedure for the Denoising Autoencoder in the picture in terms of the dataset you need and the loss function you would use. (20 Pts.)**



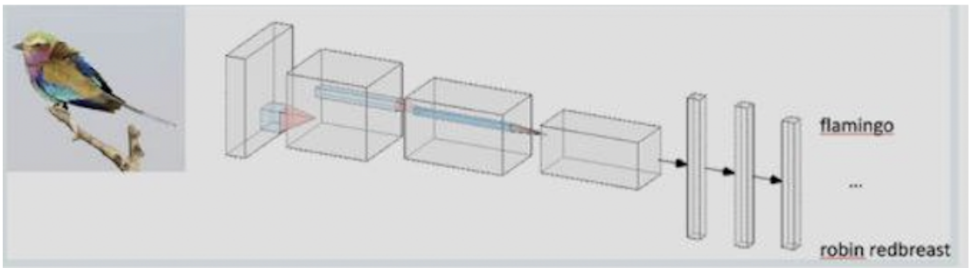
3.53. What task is addressing the network illustrated above? What loss would you use for training? How would you modify the network to also predict whether the human is an engineer or not? How would you train the network in this latter case?

3.54. Enumerate the building blocks of the above network as if you were going to implement it in Keras. Consider that there might be different viable options in terms of layers type and parameters. You can choose the one you prefer, but you need to:

- **report any assumption you make**
- **include *all* the layers and for each of them report**
  - **the layer type**
  - **the input and output sizes**
  - **the formula used to compute the number of parameters in the layer, e.g., 3x5x5=45 and not just 45.**

List each layer of the network in a different row of your answer to improve readability. (Yes you can use the calculator, but we are more interested in the formula than on the numbers!)

1) input 128x128x3 8 filtri con convoluzione 3x3 con padding same, parametri 8x(3x3x3+1) [num filtri x (3x3xD+1 bias)] output 128x128x8 2) input 128x128x8 maxpool parametri 0 output 64x64x8 2) input 64x64x8 16 filtri con convoluzione 3x3 con padding same, parametri 16x(3x3x8+1) output 64x64x16 3) input 64x64x16 32 filtri con convoluzione 3x3 con padding same, parametri 32x(3x3x16+1) output 64x64x32 2) input 64x64x32 maxpool parametri 0 output 32x32x32 4) input 32x32x32 16 filtri con convoluzione 3x3 con padding same, parametri 16x(3x3x32+1) output 32x32x16 5) input 32x32x16 flatten parametri 0 output 1x16384 6) input 1x16384 dense, parametri 16384x16+16 output 1x16 7) input 1x16 dense, parametri 16x1+1 output 1x1



The network illustrated above performs classification of different species of birds.

- What loss function would you use for training?
- How would you modify the network to perform both classification and localization of the bird? Consider that the very same network takes as input a single image and provide as output both the class and a bounding box around the bird. You can safely assume that each image contains a single bird.
- What annotations are needed to address both task in a supervised manner? What would be the training loss in this case?
- How can you modify the network to address both the classification and localization tasks when only classification labels are provided?

3.55. How can you increase the depth and the number of nonlinearities) of the above network while keeping (i) the same number of parameters in the convolutional part of the network and (i) the same receptive field? (just briefly describe the idea, don't describe the model summary) Would this be beneficial? (20/20 Pts.)

Potrei usare altri layer di convoluzione ma con filtri più piccoli, così i parametri non aumenterebbero ma avrei una rete più profonda

3.56. You don't have many images however, and you want your network to be robust to different positioning of the webcam in front of the wall clock, different weather / light conditions. What kind of data augmentation should be avoided during training? (0/1.5)

- noise addition
- ✓ vertical flip
- ✓ horizontal flip
- ✓ rotation
- change in brightness
- image scaling
- scaling of x axis
- scaling of y axis
- translation
- blur

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

input_shape = (128,256,3)

input_layer = layers.Input(shape=input_shape, name='input')
conv1 = layers.Conv2D(16, (5,5), strides=(1,1), padding='same',
        activation='relu', name='conv1')(input_layer)
mp1 = layers.MaxPooling2D(name='mp1')(conv1)
conv2 = layers.Conv2D(32, (3,3), strides=(1,1), padding='same',
        activation='relu', name='conv2')(mp1)
mp2 = layers.MaxPooling2D(name='mp2')(conv2)
dropout = layers.Dropout(0.3)(mp2)
conv3 = layers.Conv2D(128, (1,1), strides = (1,1), padding ='same',
        activation='relu', name='conv3')(dropout)
batchNorm = layers.BatchNormalization(name='batchNorm')(conv3)
        # this normalizes each slice of the volume independently
convt1 = layers.Conv2DTranspose(32, (3,3), strides=(2,2), padding='same',
        activation='relu', name='convt1')(batchNorm)
convt2 = layers.Conv2DTranspose(16, (3,3), strides=(2,2), padding='same',
        activation='relu', name='convt2')(convt1)
output_layer = layers.Conv2DTranspose(3, (1,1), strides=(1,1), padding ='same',
        activation= 'sigmoid', name='output')(convt2)
model = keras.Model(inputs=input_layer, outputs=output_layer, name='model')
```

What is the receptive field of a pixel at the center of the output of `batchNorm` layer? (1 punto)

- 9x9
- 5x5
- 12 x 12
- 8x8
- ✓ 4x4
- 6x8
- 10 x 10

3.57. Consider the network is now compiled as follows, model.compile(optimizer=tf.optimizers.Adam0, loss=categorical, and that you have a large training set of images from a surveillance camera, with whatever annotation needed. Select all the tasks for which a neural network expert (like you are expected to be) would train and use this network for: (1 punto)

- **determining how many persons appears in the image**
- **determining the image regions covered by cars, by persons and anything else**
- **determining whether it is winter or summer whether it is rain no or no**
- **determining which pixels contain a human, a car and what is the temperature in there**
- **determining where cars are parked in the image**
- **determining where there are empty parking slots**
- **tracking persons, cars and buses moving in she scene**
- **determining the location of each person, each dog, each car in the scene**
- **determining all the pixels covered by road, sky or others**

3.58. Consider the InceptionNet module. Which of the following statements are true?

✓ it uses multiple convolutional filters of different sizes in parallel
it was the first to introduce skip connections
it has been the first module used for semantic segmentation
it was the first learnable upsampling filter
✓ it leverages 1x1 convolutions to reduce the computational burden

3.59. What are GANs? "Briefly" describe their training process.

Generative adversarial network. Son composte prinopalmemte da due parti generator e discriminator. I primo is occupa di generare immagini mentre il secondo si occupa di capire so I immagine e un immagine vera o generata dal generator. Il training e fatto facendo k- times gradient descent sul discrminator, cereando di minimizzare la funzione di loss rispetto a parametri tetaD del discrimnator, Viene poi fatto gradient descent sul generator cercando di minimizzare la funzione rispetto ai parametri totaD del generator. Alla fine del training il discriminator non è più in grado di capire se l'immagine è vera o falsa (generata dal generator), così viene scartato il discriminator e viene tenuto solo il generator che può essere usato ad esempio per data augmentation.

3.60. Please briefly describe the above model (missing), indicating

- **what type of model is being trained**
- **what type of task this network is meant to solve**

Give an example of a visual recognition task that this network can be trained for and a task that this network cannot be trained for. Please, refrain from mentioning tasks that a N expert would never train this network for, even though the training function can be in principle invoked.

The model represents a convolutional neural network using residual connections, a particular type of skip connection aimed which generally aims at reducing the vanishing gradient and retain spatial information from earlier convolution layers, given that the deeper in the network that we go, the more advanced the feature get but the more spatial information is lost. The network is aimed at resolving a multi class classification task, which we can infer from the fully connected part of the network containing a softmax activation. In specific, we have 10 possible classes. We might use this kind of network to classify among species of animals, whereas we might not use it for semantic segmentation because we have a fully connected part of the network, making it incompatible.

3.61. What does the sentence "You shall know a word the company it keeps" by John R. Firth (1957) mean? Why do we mentioned it in the course and which model uses it? Describe the model

4 RNN

4.62. In Recurrent Neural Networks (RNN), what is the Vanishing Gradient due to? How this is fixed by Long Short-Term Memories (LSTM)? Why does this fix the problem? (3/3 Pts.)

Nelle RNN il vanishing gradiont è principalmento dovuto all' unfolding dei neuroni con connescioni ricorrenti, questi infatti non avendo attivazioni lineari a lungo andare possono soffrire il problema del vanishing gradient, proprio per questo le RNN di solito non riescono ad andare più di 10 step indietro nel tempo. Nelle LSTM questo problema è risolto tramite una memoria cho usa solo attivazioni lineari e pesi fissati ad 1, utilizza costant error carousel (CEC), Il contenuto della memoria viene pol letto e scritto attraverso opportuni gate. Questo risolve il problema perche andando indietro nel tempo la LSTM vede il gradiente moltiplicarsi sempre per il termine costante 1 mentre altraversa la memora

Given that in order to train a RNN we need to "unfold" the network for each time step, if the number of time steps is too big then the gradient during training gets multiplied by small (leading to either vanishing or exploding) numbers too many times leading to the vanishing gradient problem. This usually happens after if the RNN has more than 10 time steps, which means that we can't use the RNN to go back in time too much. The LSTM were employed to solve this problem by using smaller linear BLOCKS that have an input, output, memory and forget gate. LSTMs help solve the vanishing gradient because my manipulating the gradient of the forget gate, we are able to better tune the training process and avoid such problems.

4.63. Why do we need an attention mechanism? Aren't Long Short-Term Memories enough? (1 Pts.)

4.64. What is the goal of Word Embedding and what motivates it? (1/1 Pts.)

Word embedding is a task that is used in Natural Language Processing tasks. It consists in projecting the word original representation onto a latent space. Word embedding have multiple goals: reduce dimensionality (space compression) and make similar words near in the latent space. Word embedding is motivated by the fact that words are typically represented using one-hot encoding over a dictionary. It means that each word will have all elements equal to zero and only one 1: extremely sparse representation. Moreover, two similar words will be equally distant independent on how much they are similar. However, when we analyse text for several tasks (sentiment analysis, word prediction...), we would like to be able to represent words in a small and dense space being able to directly compute how much two words are similar. These are the reason why word embeddings are used.

4.65. Can word embedding overfit? Motivate your answer and, in case it does, suggest a way to detect it. (20 Pts.)

Yes, word embedding can overfit as all other machine learning and deep learning tasks. If a word embedding overfits, it means that it has not learnt to represent the words and to learn a latent space. It will have simply learnt to reproduce the words in input. It is possible that the word embedding overfits because of the complex architectures we are likely to use to perform this task, which somehow replicates the reasoning of an autoencoder (it has to learn an embeddina). We can overfit if we do not validate our data or even if we have an extremely complex network to perform the word embeddings. We can detect if a model is performing overfitting on word embeddings by using cross validation techniques. In this way, the model will be tested on new unseen samples.

Theoretically yes. For example, if we use a latent representation that is big as the vocabulary we risk overfit. Another example is if we use lots of hidden (that are prone to overfitting) we could risk overfitting the word embedding. Concerning word2vect, which is based on a few layers, I think that this become a rare possibility since our vocabulary is generally much higher than the N-words that are used for context handling. Theoretically yes, practically not common

4.66. When we could prefer the use of Recurrent Neural networks instead of Long Short-Term Memory networks? Why? (10 Pts.)

4.67. What is an STM and why it helps with vanishing gradient? (20 Pts.)

4.68. What is word embedding? What it is used for?

4.69. Describe the Word2Vec network architecture. (2/2 Pts.)

Word2Vec is an architecture proposed by google with two different possible implementations: Bag Of Words and SkipGram. Word2Vec is a model working on sentences of words to be able to perform natural language processing tasks such as Masked Language Modelling (detect what is the missing word given its context). The former implementation takes the context of a word (the words near the target word) and computes which is the most likely word to be in that position. SkipGram implementation takes a word as input and outputs a vector of probabilities representing the probabilities of each position of the context to represent a word: it means it computes for each position a vector of probabilities. In that way, it computes the most likely word in each position. Word2Vec implementation uses matrices: it has one matrix that is used to project the input on the words representation and another matrix to pass from this representation to the vector (or the) probbilities (probability). In case of BOW the context words are projected onto the same vector and are averaged

4.70. How does word2vect work? (0/10 Pts.)

4.71. Do you use word embedding with LSTM networks? How? (0/10 Pts.)

4.72. Do you use word embedding with transformer? How?

4.73. Text is a challenging domain where several challenges need to be faced to learn successful models. With reference to machine learning on text data answer the following questions:

- **Text requires to be encoded, describe what is word embedding, what it is used for, and how it is obtained with the word2vec model.**
- **Text comes in sequences, describe the Long Short-Term Memory cell and the architectures which can be used when inference is done online, i.e., one word at the**

time, and batch, i.e., when the entire sentence is available.

- **Long Short-Term Memory cells solve the vanishing/exploding gradient problem of Recurrent Neural Networks. What is this problem due to? How do they solve it?**

4.74. Do I really need a recurrent neural network to process a stream of input data? Why can't I use a standard feed forward architecture? (10 Pts.)

4.75. How does the WRITE mechanism works in a Neural Touring Machine? (10 Pts.)

4.76. How does the READ mechanism work in a Neural Touring Machine? (10 Pts.)

4.77. What is the difference between a Recurrent Neural Network and a Long Short-Term Memory in terms of architecture? (Try to be short and focused!) (20 Pts.)

4.78. What is the difference between a Recurrent Neural Network and a Long Short-Term Memory in terms of training algorithms? (Try to be short and focused!) (20 Pts.)

4.79. How data representation is learned via deep autoencoders? (20 Pts.)

4.80. How data representation is leaned via recurrent neural network? (20 Pts.)

4.81. What is the pain behind text encoding? What is the relief? (20 Pts.)

4.82. How could we size the embedding of a neural autoencoder? (1 Pts.)

4.83. Why do we need recurrence mechanism? Isn't attention mechanism enough? (1 pts.)

4.84. The current state of the art in text modeling and prediction is the Transformer. Is it implemented with a Recurrent Neural Network? Does it suffer the Vanishing Gradient problem? Motivate your answer. (20 Pts.)

4.85. Can autoencoder overfit? Motivate your answer and, in case they do, explain how to detect it.

4.86. How sentences are embedded in Seq2seq modeling? (0/1 Pts.)

In a Seq2seq model words are embedded by assigning a certain integer number to each word present in the learning dictionary. We also need special characters like the <BOS> Begin of sequence, «EOS> End of sequence, <UNK> for unknown words not present in my dictionary, <PAD> because deep learning algorithm are trained in mini batches and those have to be of the same length.

Sentences are embedded using sentence embeddings. A simple approach we can use to embed a sentence (thus to produce a sentence embedding) is to compute the word embedding of each word and use a vector of word embeddings. Seq2seq models analyse sentences using an encoder-decoder architecture. The input sentence is analysed from the encoder producing the hidden states. Some characters are inserted to represent key location in text as an example a special character can be used to represent that the sentence is finished and the decoder must start to produce the output. Then, once the encoder finishes to evaluate the sequence in input, the decoder will start to analyse the sequence outputted by the encoder in an autoregressive manner. The decoder will use the output of the encoder to produce the first output, than, it will be concatenated to the input and it will be feeded to it for the next step.

Seq2seq modelling uses an encoder/decoder architecture and the embedding of sentences happens in the encoder. The encoder is usually made of recurrent cells (RNNs, LSTMs or GRUs) so to capture the sequential aspect of the problem.

4.87. Exercise 4 (6 Pts.): Consider the problem of sequence modeling and the classical Seq2seq model used for machine translation. Answer the following questions:

- **What is "sequence to sequence modeling"?**
- **Describe the Long Short-Term Memory cell and its use in sequence modeling**
- **Describe the classical Seq2seq model based on STM cells for language translation, its training procedure and its run-time inference**
- **What is an attention mechanism? How can it be used in a Seq2seq model?**

4.88. What are Neural Turing Machines? How does attention work in this kind of models? (2/2 Pts.)

Neural turing machine sono RNN che hanno anche a disposizione una memona costituita da vettori. L'RNN svolge il ruolo di controller performando operazioni di scrittura e lettura in memoria. Per rendere le operazioni di lettura e scrittura differenziabli ogni volta viene letta o scritta tutta la memoria, solo in misura diversa e qui entra in gioco l'attention mechanism che appunto indica in che misura scrivere nelle varie. Attention mechanism diviso in due fasi content-base e location-based. Nel content-based la stato attuale viene confrontato con il contenuto della memoria tramite dot product e poi viene applicata softmax per avere un primo attention vector che successivamento viene confrontato con l'attention vector dello step precedente. Nel location-based i due venoono interpolati e viene fatta una convolve tra il vattore finora ottenuto e uno shift filter con successivo sharpen, permettendo alla rete di concentrarsi anche su celle diverse.

4.89. What is the Transformer? How does attention work in this model? (2/2 Pts.)

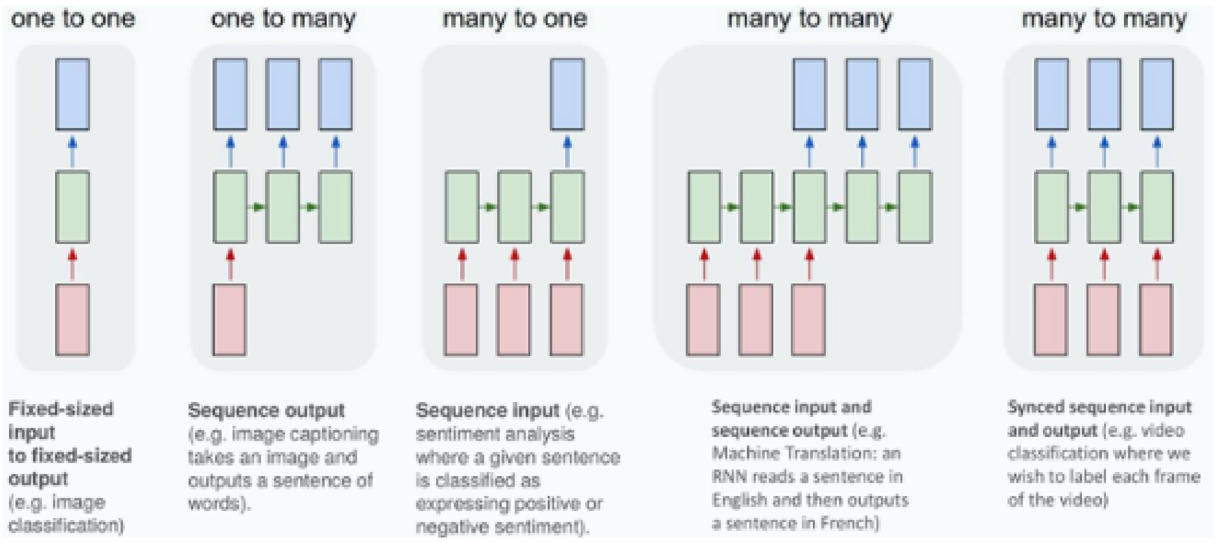
I trasformer è costituito da uno stack di encoder e da uno stack di decoder. Entrando piu nel dettaglio ogni encoder è composto da un modulo di self-attention e da una FFNN. Invece il decoder ha, oltre i due layer dell'encoder, anche un layer di encoder-decoder attention in cui riceve l'output del top encoder. Il ruolo principle è svolto dal self-attention che si occupa di identificare le parole della frase che più hanno importanza per la parola che si sta analizzando in questo momento. Una multi-head attention permette di guardare più parole e di tenere in considerazione più representation spaces. Attention calcolata tramite matrici Q,K,V rispettivamente query, key e value. Il trasformer non usa RNN e al suo posto per capire l'ordine delle parole nella frase usa un positon encoding. Il primo encoder riceve l'input dopo che è stato applicato su di esso un embedding.

4.90. How does the attention mechanism work?

The attention mechanism has been introduced to be able to focus on different parts of the input. The attention mechanism is used in several models: Neural Turing Machines, Seq2seq models, Transformers and so on. The idea is to overcome the limits imposed by previous models such as STMs and it is built on top of RNN. We aim at modelling which is the input that is more correlated with the output to increase the performance of the models that we are going to produce. In fact, in many tasks such as natural language, a word may be more correlated with some word in the past more than on the last 3/4 word, as an example. Attention in NTMs: For instance, in NTMs we use the attention mechanisms to be able to read at different locations of the memory with different extents and to write at different locations of memory at different extents (using two types of attentions called content-based attention and location-based attention). Attention in Seg2Seg: In Seg2Seg models we compute the attention using both the hidden states and the source states to produce attention scores (multiplicative or additive style) which will be used to produce attention weights. Then, we will build a context vector and an attention vector at the end. This procedure allows us to understand which are the most important inputs characterizing what we will have in output.

4.91. Describe briefly the Seq2seq model. Can it be used with RNN or it can only be used with LSTM? Why? (20 Pts.)

4.92. For each of the models in the IMAGE below provide its description and make an example of it.



one to one
one to many
many to one
many to many
many to many
Fixed-sized input to fixed-sized output (e.g. image classification)
Sequence output (e.g. image captioning takes an image and outputs a sentence of words).
Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).
Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French)
Synced sequence input and output (e.g. video classification where we wish to label each frame of the video)