**26/01/2023**

**Vanishing and exploding gradients are issues of Neural Network Training. For each of the following statements, you have to state if it is True or False.**

1. Shortcut (or skip) connections are used to face the vanishing and exploding gradients.
   True. Shortcut connections, also known as residual connections, are used to address the problem of vanishing and exploding gradients in deep neural networks. These connections allow gradients to bypass one or more layers in the network, which can help to preserve the gradient information and make it easier for the model to learn. This technique is known as Residual learning or ResNet which is used to train very deep neural networks successfully.
2. The vanishing gradient is present just in recurrent neural networks (RNN).
   False. Vanishing gradients can occur in both feedforward neural networks (FFNN) and recurrent neural networks (RNN) especially when the network is deep. In RNNs, the problem is more pronounced because the gradients must pass through many time steps, which can cause the gradients to become very small.
3. Batch normalization has been introduced to solve the vanishing gradient issue.
   False. Batch normalization is a technique that is used to reduce the internal covariate shift and to stabilize the training of deep neural networks. It does this by normalizing the activations of each layer to have zero mean and unit variance. Batch normalization does not directly address the issue of vanishing gradients, but it can make the network more robust to the initialization of the parameters, which may allow the gradients to propagate more easily through the layers.
4. Rectified Linear Unit (i.e., ReLUs) are useful in mitigating exploding gradient but not vanishing gradient.
   False. ReLU (Rectified Linear Unit) activation function, which outputs the input if it is positive, and 0 otherwise, can solve both issues. Indeed, the derivative of ReLU is 1, besides when there is a dead neuron issue, this helps in both cases.
5. Rectified Linear Unit (i.e., ReLUs) are useful in mitigating vanishing gradient but not exploding gradient.
   False. ReLU (Rectified Linear Unit) activation function, which outputs the input if it is positive, and 0 otherwise, can solve both issues. Indeed, the derivative of ReLU is 1, besides when there is a dead neuron issue, this helps in both cases.
6. The exploding gradient is solved in long short-term memories (LSTM).
   True. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are designed to overcome the vanishing gradient problem by introducing a memory cell, gates, and a cell state. These components allow the LSTM to selectively store and access information, and to prevent gradients from exploding or vanishing during backpropagation.

**You have to train a feedforward neural network with a limited amount of data and you want to use as much as possible of these. Describe how you could train the model to prevent overfitting and motivate your choices in light of the previous statement.**

The core of this question is on the reduced amount of data so you should use techniques to limit overfitting which do not require the hold out of data as much as possible. Proper methods are dropout and weight decay, while early stopping requires you to waste data and the exercise was exactly about avoiding this!

For hyperparameter tuning, e.g., for the lambda in weight decay, k-fold/holdout/loo can be used, but once the proper parameter has been selected all data should be put back together and the model should be retrained. Also, Data Augmentation and Transfer Learning could be used to face the problem of limited data.

**Before the invention of Transformers, seq2seq models with attention where the state of art in neural language translation.**

**a) Describe the seq2seq model without attention**

**b) How is attention added to seq2seq models? Why it was added?**

**c) What is word2vec? How it is related to seq2seq models?**

a) A seq2seq model without attention is a type of neural network architecture that is used for tasks such as language translation, text summarization, and image captioning. The basic architecture consists of two main parts: an encoder, which takes in a sequence of input data and produces a fixed-length context vector, and a decoder, which takes the context vector and produces a sequence of output data. The encoder and decoder are typically implemented using recurrent neural networks (RNNs) such as LSTMs or GRUs. The encoder's job is to encode the input sequence into a single vector, called a context vector, which contains information about the entire input sequence. The decoder then uses this context vector to generate the output sequence. Training is performed by providing the proper sequence to the decoder (teacher forcing) while at inference time an autoregressive approach is used providing as input the word just predicted by the decoder.

b) Attention is added to seq2seq models to improve the ability of the model to focus on specific parts of the input sequence when generating the output sequence. Attention mechanisms work by allowing the decoder to "attend" to different parts of the input sequence at different time steps, rather than using a fixed context vector. Attention mechanisms are implemented by introducing a new component, called an attention mechanism, that learns to assign weights to different parts of the input sequence at each time step. The attention mechanism is typically implemented using a feedforward neural network, which takes as input the current decoder state and the encoder states and produces a set of attention weights. These attention weights are then used to weigh the encoder states when generating the next output. Attention was added to seq2seq models to improve the performance on tasks that require the model to focus on specific parts of the input when generating the output, such as language translation and text summarization.

c) Word2vec is a technique for learning vector representations of words, also known as word embeddings. It is a neural network-based technique that learns to predict the context of a given word based on its surrounding words. Word2vec is related to seq2seq models in the sense that it is often used as a pre-training step for the encoder in a seq2seq model. The embeddings learned by word2vec can be used to initialize the embedding layer of the encoder, which can improve the performance of the seq2seq model on tasks such as language translation and text summarization.

**Assume you have to train a CNN that**

**- takes as input a grayscale picture of a binary number as the one below, and**

**- predicts the value of the encoded number**

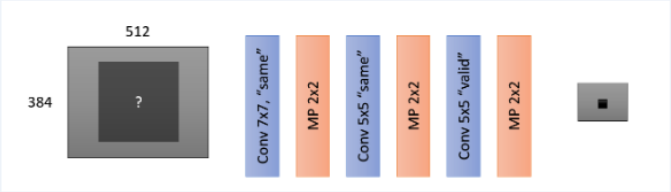

**Please mark all the sentences that are correct.**

**Scegli una o più alternative:**

a. Normalizing the data before training can be beneficial. This is true, like other inference problems on images.
b. I can perform data augmentation by zooming, resizing and resampling the input image at training time. True but no explanation
c. It is convenient to train the network minimizing categorical cross entropy, as long as the number to be predicted are positive.
d. I can perform data augmentation by translating the image and by minor perspective changes. Sure, these should not change the associated label.
e. I can perform data augmentation by vertical and horizontal flips and by adding moderate amount of blur and noise.
f. I can select a portion of data as validation set and use this to assess stopping criteria. Sure, this is a good practice to have a validation set to assess stopping criteria.

**Receptive Field**

What is the receptive field at the central location of the activation map after the followign layers?




Scegli un'alternativa:

- a. 42x42
- b. 24x24
- c. 28x42
- d. it is not possible to say since the output size are not defined
- e. 38x38 ✔

## GANs Training

The following questions concern the GANs training process.

Please select all the correct statements.

Scegli una o più alternative:

- a. Once the training converge, the Discriminator can be discarded. ✔ This is true, all that we need after training the GAN is the Generator.
- b. GANs' training is rather an unstable process, still it is possible to train a GAN by standard tools like backpropagation and some regularization like dropout. ✔ True, it is unstable because we need to alternate Generator and Discriminator training. The first GAN in 2014 actually were trained by backpropagation and regularization by dropout.
- c. GANs is made of two networks performing classification but that are trained pursuing opposite goals
- d. GANs are neural networks originally designed for generating images. ✔ True, that's their main purpose
- e. GANs can be used to generate human faces of high resolution.
- f. At the begining the Generator and the Discriminator are randomly initialized and there is no need to update the Generator untill the Discriminator converges.

Le risposte corrette sono:

GANs are neural networks originally designed for generating images.

GANs' training is rather an unstable process, still it is possible to train a GAN by standard tools like backpropagation and some regularization like dropout.,

Once the training converge, the Discriminator can be discarded.,

GANs can be used to generate human faces of high resolution.

## Architecture

Consider the following code

```
import tensorflow as tf
tfkl = tf.keras.layers
tfk = tf.keras
def get_model_1(input_shape, num_classes):
    input_layer = tfkl.Input(shape=input_shape, name='input_layer')
    cd1 = tfkl.Conv2D(128, kernel_size=3, padding='same', activation='relu', name='conv_d1')(input_layer)
    d1 = tfkl.MaxPooling2D(name='mp_d1')(cd1)
    cd2 = tfkl.Conv2D(256, kernel_size=3, padding='same', activation='relu', name='conv_d2')(d1)
    d2 = tfkl.MaxPooling2D(name='mp_d2')(cd2)
    cd3 = tfkl.Conv2D(512, kernel_size=3, padding='same', activation='relu', name='conv_d3')(d2)
    u1 = tfkl.UpSampling2D(name='up_u1')(cd3)
    u1 = tfkl.Concatenate(name='concat1')([u1,cd2])
    u1 = tfkl.Conv2D(256, kernel_size=3, padding='same', activation='relu', name='conv_u1')(u1)
    u2 = tfkl.UpSampling2D(name='up_u2')(u1)
    u2 = tfkl.Concatenate(name='concat2')([u2,cd1])
    u2 = tfkl.Conv2D(128, kernel_size=3, padding='same', activation='relu', name='conv_u2')(u2)
    output_layer = tfkl.Conv2D(num_classes, kernel_size=3, padding='same', activation="softmax", name='output_layer')(u2)

    model = tf.keras.Model(inputs=input_layer, outputs=output_layer)
    model.compile(
        loss = tfk.losses.SparseCategoricalCrossentropy(),
        optimizer = tfk.optimizers.Adam(),
        metrics = [
            tfk.metrics.Accuracy(),
            tfk.metrics.MeanIoU(num_classes=num_classes)
            ]
        )
    return model
```

```
input_shape = (128,256,3)
num_classes = 3
model = get_model_1(input_shape=input_shape, num_classes=num_classes)
s=model.summary()


The last command returns the following summary, which we provide for reference
_________________________________________________________________________________________________
 Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
 input_layer (InputLayer)
 conv_d1 (Conv2D)
 mp_d1 (MaxPooling2D)
 conv_d2 (Conv2D)
 mp_d2 (MaxPooling2D)
 conv_d3 (Conv2D)
 up_u1 (UpSampling2D)
 concat1 (Concatenate)
 conv_u1 (Conv2D)
 up_u2 (UpSampling2D)
 concat2 (Concatenate)
 conv_u2 (Conv2D)
 output_layer (Conv2D)
=================================================================================================
Total params:
Trainable params:
Non-trainable params:
_________________________________________________________________________________________________
```

**Solution:**

```
 Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
 input_layer (InputLayer)     [(None, 128, 256, 3)]     0           []
 conv_d1 (Conv2D)             (None, 128, 256, 128)     3584        ['input_layer[0][0]']
 mp_d1 (MaxPooling2D)         (None, 64, 128, 128)      0           ['conv_d1[0][0]']
 conv_d2 (Conv2D)             (None, 64, 128, 256)      295168      ['mp_d1[0][0]']
 mp_d2 (MaxPooling2D)         (None, 32, 64, 256)       0           ['conv_d2[0][0]']
 conv_d3 (Conv2D)             (None, 32, 64, 512)       1180160     ['mp_d2[0][0]']
 up_u1 (UpSampling2D)         (None, 64, 128, 512)      0           ['conv_d3[0][0]']
 concat1 (Concatenate)        (None, 64, 128, 768)      0           ['up_u1[0][0]', 'conv_d2[0][0]']
 conv_u1 (Conv2D)             (None, 64, 128, 256)      1769728     ['concat1[0][0]']
 up_u2 (UpSampling2D)         (None, 128, 256, 256)     0           ['conv_u1[0][0]']
 concat2 (Concatenate)        (None, 128, 256, 384)     0           ['up_u2[0][0]', 'conv_d1[0][0]']
 conv_u2 (Conv2D)             (None, 128, 256, 128)     442496      ['concat2[0][0]']
 output_layer (Conv2D)        (None, 128, 256, 3)       3459        ['conv_u2[0][0]']
=================================================================================================
Total params: 3,694,595
Trainable params: 3,694,595
Non-trainable params: 0
```

**Mark all the tasks that the above network can be trained for.**

**Please, refrain from marking answers that are "technically possible in principle in Python", and stick to those tasks that a Neural Network expert (as you are expected to be) would use this network for.**

**Scegli una o più alternative:**

a. Weather forecasting
b. Given an input grayscale image, bring this back to colors
c. Counting the number of cars passing on a street.
d. Determining which pixels in an image belongs to road, sky and anything that does not belong to these classes
e. Determining which pixels in an image of a checkerboard are white, which are black and which are covered any piece.
f. Given an histological image, determining pixels covered by cells, tubules and interstitial area between cells.
g. Determining which pixels in an image belongs to car and which belongs to other.

It's a 3 class segmentation problem. The correct answers are:

Determining which pixels in an image of a chekerboard are white, which are black and which are covered any piece.,

Determining which pixels in an image belongs to road, sky and anything that does not belong to these clasess.,

Given an histological image, determining pixels coverd by cells, tubules and interstitial area between cells.