Paula Castillo, 184646
Helena Cots, 183951

PRÀCTICA 4: "Deep Learning"

Part1: Feature extraction with Alexnet

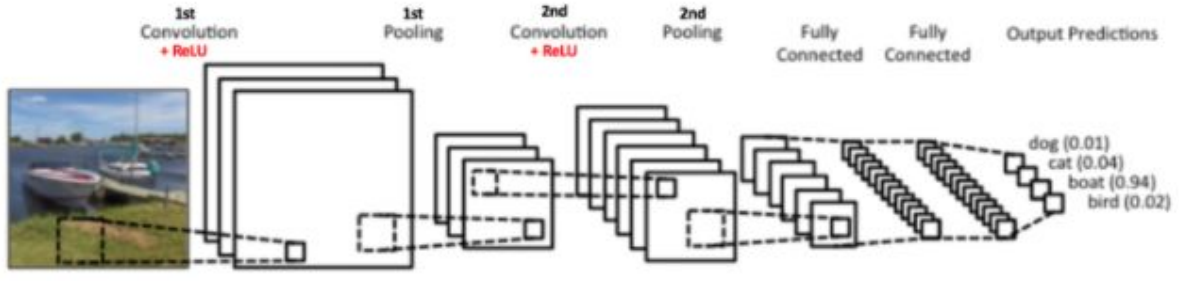
Load the pre-trained Alexnet Network and take a look at its structure using the command net.Layers. Enumerate the different kind of layers and explain them.

```
25x1 Layer array with layers:

   1   'data'     Image Input                  227x227x3 images with 'zerocenter' normalization
   2   'conv1'    Convolution                  96 11x11x3 convolutions with stride [4  4] and padding [0  0]
   3   'relu1'    ReLU                         ReLU
   4   'norm1'    Cross Channel Normalization  cross channel normalization with 5 channels per element
   5   'pool1'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
   6   'conv2'    Convolution                  256 5x5x48 convolutions with stride [1  1] and padding [2  2]
   7   'relu2'    ReLU                         ReLU
   8   'norm2'    Cross Channel Normalization  cross channel normalization with 5 channels per element
   9   'pool2'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
  10   'conv3'    Convolution                  384 3x3x256 convolutions with stride [1  1] and padding [1  1]
  11   'relu3'    ReLU                         ReLU
  12   'conv4'    Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1]
  13   'relu4'    ReLU                         ReLU
  14   'conv5'    Convolution                  256 3x3x192 convolutions with stride [1  1] and padding [1  1]
  15   'relu5'    ReLU                         ReLU
  16   'pool5'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0]
  17   'fc6'      Fully Connected              4096 fully connected layer
  18   'relu6'    ReLU                         ReLU
  19   'drop6'    Dropout                      50% dropout
  20   'fc7'      Fully Connected              4096 fully connected layer
  21   'relu7'    ReLU                         ReLU
  22   'drop7'    Dropout                      50% dropout
  23   'fc8'      Fully Connected              1000 fully connected layer
  24   'prob'     Softmax                      softmax
  25   'output'   Classification Output        crossentropyex with 'tench', 'goldfish', and 998 other classes
```

Tenim 3 tipus de capes en la xarxa neuronal després del *input*: **convolució → RELU → Max-Pooling**, que es repeteixen fins a obtenir una xarxa completament connectada (**fully connected layer**) i arribar a la capa de classificació.

Aquestes capes són els elements bàsics per a la construcció dels blocs de qualsevol CNN. Juntes, extreuen característiques útils de les imatges, introdueixen no-linearitat a la xarxa i redueixen la dimensió de les característiques.

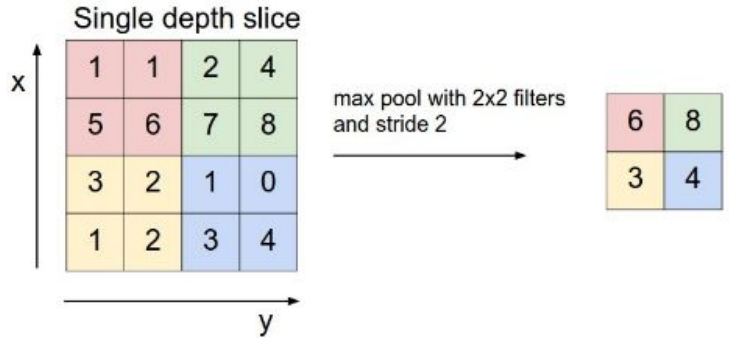


- Image **Input** Layer: és on especifiquem la mida de la imatge (alçada i amplada) i la mida del canal. La “digit data” consisteix en imatges en escala de grisos pel que la mida del canal de color és 1. Per una imatge en color la mida del canal és 3.

- **Convolutional** Layer: aquesta capa és l’encarregada de computar la sortida de les neurones, que es corresponen amb petites regions locals de l’“input”. Per a fer això, calcula el producte escalar entre aquesta petita regió i els seus pesos. Seguidament, s’aplica un altre filtre, i per això se li passa com a argument la mida d’aquest. Com a segon argument se li passa la quantitat de filtres que tenim, que és el número de neurones que connecten a la mateixa regió del input. A més, aquest paràmetre determina el número de mapes de característiques. Finalment, utilitza la funció de “padding” per a afegir padding al “input feature map”, i li passem com a arguments “padding” i la quantitat que en volem.

- **Relu** Layer: ReLU significa “rectified linear unit” i és una functió d’activació no-linear.

- **maxPooling** Layer: realitza una operació de downsampling al llarg de les dimensions espacials resultant en una reducció d’aquestes.Així, aconsegueix reduir la quantitat de paràmetres i informació redundant en la xarxa, i controlar també l’“overfitting”. El que retorna aquesta capa són els valors màxims de les regions d’entrada rectangulars, la seva mida s’especifica a *poolSize*.

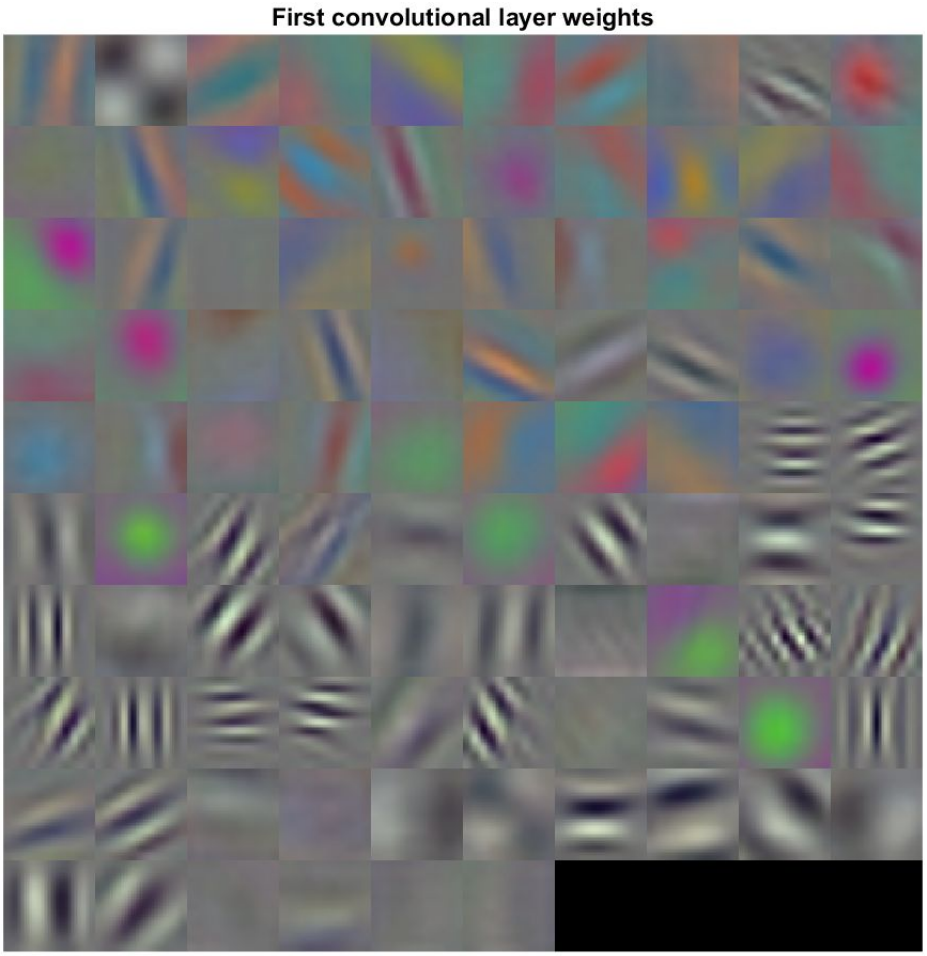


- **Fully Connected** Layer: la capa de downsampling va seguida d’una o més “fully connected layer”. Com el seu nom indica, és una capa on el conjunt de neurones es connecten amb totes les neurones procedents de les altres capes. Combina les característiques extretes a partir de les altres capes amb la imatge per a identificar els patrons més destacats. L’última *fully connected layer* combina totes les característiques amb el fi de classificar les imatges.

- **Softmax function**: normalitza l’output de la capa prèvia obtenint així una seqüència de valors positius que sumen 1, tals que poden fer servits per a classificar.

- **Classification Layer**: aquesta capa és la última en actuar. Utilitza les probabilitats obtingudes a la capa anterior per a assignar a cada *input* a una de les classes i calcular l’error.

Plot the weights of the first convolutional layer as it is done in the example and explain intuitively which kind of features do you think that the network is extracting.



En la imatge que veiem a dalt, es veuen representats els pesos de la primera capa convolucional. Cada una de les capes convolucionals extreu unes característiques determinades, cosa que ens recorda a un filtre anomenat filtre de Gabor: extreu característiques concretes, com poden ser la orientació de les línies, la seva amplada, els colors, etc.

La primera capa ha utilitzat filtres per capturar la massa amorfa i els contorns de les imatges d'entrada. Aquests pesos es consideren les característiques primàries que conformen la imatge.

As you can see in the parts Extract Training Features Using CNN and Train A Multiclass SVM Classifier Using CNN Features, CNNs can be used to extract features from your images and then use this features to train a classifier such as a SVM. Do this with the CKBD dataset and report the accuracy obtained. (compute the accuracy).

Hem utilitzat les característiques d'imatge de la Xarxa Neuronal Convolucional de Alexnet per entrenar el classificador SVM per a multi-classes.

Com podem veure a les capes de la xarxa neuronal, la 'fc7' correspon a una 'capa totalment connectada' que conté tot el conjunt de característiques primitives combinades en una representació rica de la imatge. És aquesta capa la que es fa servir per tenir en compte les característiques de la nostra base de dades d'imatges, ja que és la més adequada per a tasques de reconeixement. Utilitzem el mètode *'activations*' per a extreure les característiques d'aquesta capa. Amb això, obtenim les noves 'dades' amb les característiques d'entrenament corresponent, que es divideixen entre les que es faran servir per entrenament i per test.

Utilitzem la funció *fitcecoc* per classificar aquestes dades amb un classificador SVM multi-classes. Posteriorment fem les prediccions corresponents i computem l'accuracy.

Amb aquesta CNN obtenim una accuracy d'entre 85 i 90 %.

Part 2: CNN classifier

Try different configurations of layers and analyse the results, try to get an accuracy as high as possible! You can choose how many convolutional layers do you want to use and with which parameters and activation functions, which other layers will you apply... There is not a correct way to do that, but you should understand well what each layer is doing in order to have some intuition about what could be useful.

Expliquem breument el tipus de capes que tenim a la nostra xarxa CNN i els seus paràmetres, i que modifiquem per augmentar l'accuracy de la nostra classificació. Posteriorment, expliquem com les hem modificat o quines i perquè ho hem decidit.

- Image **Input** Layer: com s'ha explicat a la part 1, en aquesta capa li entra la mida de la imatge i la del canal. En el nostre cas, al tractar-se d'imatges en escala de grisos, la mida del canal és 1. [*height, width, 1*]
- **Convolutional** Layer: el primer argument és la mida del filtre (alçada i amplada); el segon identifica el nombre de filtres que tenim, com s'ha explicat prèviament corresponen al nombre de neurones que estan connectades en una mateixa regió del *input*. Podem utilitzar el paràmetre 'Padding' i especificant la mida d'aquest. [*filterSize, numerOfFilters, 'Padding', paddingSize*]
- **Batch Normalization** Layer: aquesta capa normalitza les activacions i els gradients propagant-se per una xarxa, fent que l'entrenament d'aquesta xarxa sigui més òptim. S'utilitza juntament amb la capa ReLU per a augmentar la velocitat de l'entrenament i reduir la sensibilitat de la inicialització de la xarxa.
- **Relu** Layer: ReLU significa "rectified linear unit" i és una functió d'activació no-linear.
- **maxPooling** Layer: com que retorna els màxims d'una regió del input en concret per a reduir dimensionalitat, com a primer paràmetre té la mida del rectangle del input d'on s'agafarà el màxim, *poolSize*. En aquest cas, el segon argument és 'Stride', seguit de la seva mida, que especifica la mida del pas que la funció d'entrenament utilitza mentre escaneja al llarg del input. [*poolSize, 'Stride', strideSize*]
- **Fully Connected** Layer: en aquesta capa totes les neurones connecten amb totes les altres neurones en la capa precedent. Combina les característiques apreses per la capa anterior a través de la imatge per identificar patrons més grans. La última capa "Fully-Connected" combina totes les característiques per a classificar les imatges. El paràmetre de la mida de sortida d'aquesta capa equival al nombre de classes en què es classifiquen les dades, en el nostre cas és el nombre d'emocions usades. [numberOfClasses]
- **Softmax** Layer: normalitza la sortida de la capa totalment connectada. La sortida d'aquesta funció consisteix en un conjunt de nombres positius que sumen 1, que poden ser usats com a les probabilitats de la classificació de la següent capa: la capa de classificació.
- **Classification** Layer: l'última capa és la capa de classificació. Aquesta, utilitza les probabilitats retornades per la funció d'activació *softmax* per a cada entrada per assignar l'entrada a una de les classes excloents i calcular la precisió.

Explain in detail your final CNN, their layers, and how and why have you arrived to that configuration. Report the accuracy of your model and compare it with the one obtained in part 1.

Mostrem un parell d'exemples previs a arribar a la configuració de capes que ens ha donat millors resultats (n'hem provat moltíssimes).

```
31
32          convolution2dLayer(3,16,'Padding',1)
33          batchNormalizationLayer
34          reluLayer
35
36          maxPooling2dLayer(4,'Stride',2)
37
38          convolution2dLayer(3,32,'Padding',1)
39          batchNormalizationLayer
40          reluLayer
41
42      %       maxPooling2dLayer(2,'Stride',2)
43      %
44      %       convolution2dLayer(3,64,'Padding',1)
45      %       batchNormalizationLayer
46      %       reluLayer
47
48          fullyConnectedLayer(7)
49          softmaxLayer
50          classificationLayer];
51
52      %%
53 -    frequency=4; %30 default
```

Command Window

```
accuracy =

    0.7358
```

```
29 -    layers = [
30          imageInputLayer([84 84 1])
31
32          convolution2dLayer(3,16,'Padding',1)
33          batchNormalizationLayer
34          reluLayer
35
36          maxPooling2dLayer(2,'Stride',2)
37      %
38          convolution2dLayer(5,32,'Padding',1)
39          batchNormalizationLayer
40          reluLayer
41
42          maxPooling2dLayer(2,'Stride',2)
43
44          convolution2dLayer(6,64,'Padding',1)
45          batchNormalizationLayer
46          reluLayer
47
48          fullyConnectedLayer(7)
49          softmaxLayer
50          classificationLayer];
51
```

Command Window

```
accuracy =

    0.7547
```

Hem anat provant diverses configuracions de la nostres capes per a tal de saber si en necessitàvem més d'una o no. Podríem dir que depenent dels paràmetres que passem a les funcions, a vegades calen menys o més capes. és a dir, si col·loquem més filtres a la primera i segona capa convolucional potser no ens cal tenir una tercera d'aquestes. Tot i així, ens ha donat sempre millor resultat quan variem els paràmetres dels 3 blocs de capes.

A la primera capa convolucional creiem que és important que l'estructura dels filtres no sigui molt gran ja que així podrà agafar amb més precisió les característiques del *input*. A les capes on es fa el *maxpooling* agafem elements estructurals no molt grans, que ens permetin reduir la dimensió dels mapes de característiques però alhora sense perdre gaire informació. Per això agafem el màxim d'entre 4 valors [2x2].

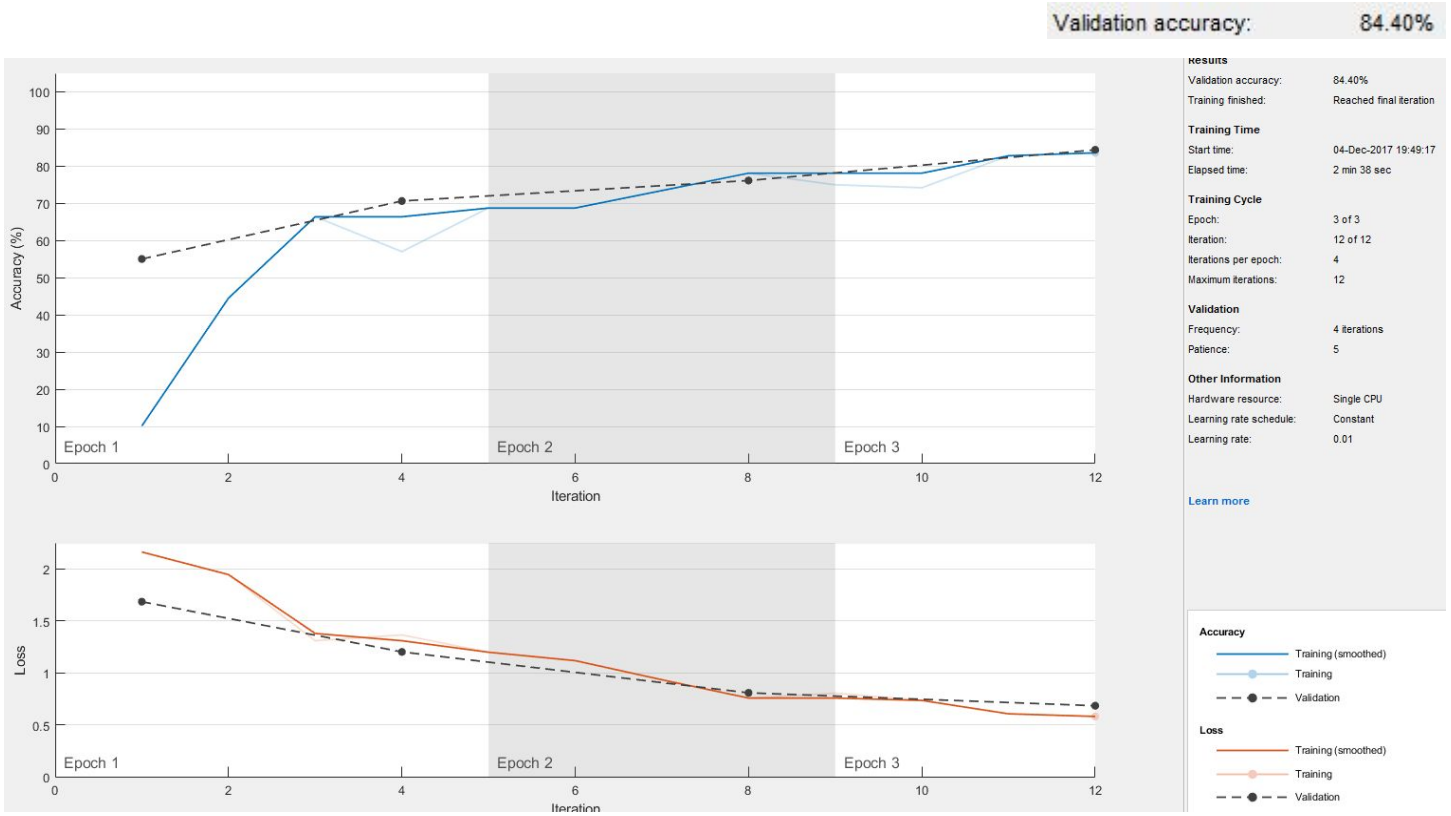
Hi hem col·locat tres capes convolucionals. A cada una es va augmentant la mida del filtre i el nombre de filtres, així és com ens ha donat la millor precisió. La quantitat de filtres augmenten exponencialment: 16,32,64. I la mida augmenta, respectivament, com 3x3, 5x5, 14x14. Pot ésser que no hi hagi un valor de mida correcte per a aquests filtres, però són els que ens han donat millors resultats.

Mostrem la configuració final de les nostres capes:

accuracy	0.8440
ans	[636 128 128]
emotionsUsed	[0 1 2 3 4 5 6 7]
frequency	4
imagesData	*4-D double*
indexes	*654x1 double*
k	1
labels	*654x1 categorical*
layers	*15x1 Layer*
net	*1x1 SeriesNetwork*
options	*1x1 TrainingOptionsS...*
predictedLabels	*109x1 categorical*
shapeData	*654x68x2 double*
stringLabels	*654x9 char*
valLabels	*109x1 categorical*
Xtest	*4-D double*
Xtrain	*4-D double*
Ytest	*109x1 categorical*
Ytrain	*545x1 categorical*

```
27      %%
28 -    layers = [
29          imageInputLayer([84 84 1])
30
31          convolution2dLayer(3,16,'Padding',1)
32          batchNormalizationLayer
33          reluLayer
34
35          maxPooling2dLayer(2,'Stride',2)
36      %
37          convolution2dLayer(6,32,'Padding',1)
38          batchNormalizationLayer
39          reluLayer
40
41          maxPooling2dLayer(2,'Stride',2)
42
43          convolution2dLayer(14,64,'Padding',1)
44          batchNormalizationLayer
45          reluLayer
46
47          fullyConnectedLayer(8)
48          softmaxLayer
49          classificationLayer];
```

En el plot següent veiem que ens resulta en un 84.40% la precisió de validació.



És evident que per arribar a l'accuracy que ens donava a la primera part encara ens caldria millorar la nostra configuració de les capes, ja que amb la xarxa d'entrenament *Alexnet* podíem obtenir fins a una precisió del 90%, amb aquesta en prou feines hem pogut arribar al 85%. Malauradament, caldria poder veure quines característiques extreu cada capa convolucional per a poder entendre millor com organitzar les diferents capes.

Plot the different filters of the first convolutional layer as you have done in part 1 and comment the results.

Podem visualitzar els pesos de la primera capa convolucional, però, al contrari que passava amb la capa de *Alexnet* que veiem perfectament quines característiques extreia el filtre, aquí és força més complicat d'entendre-ho.