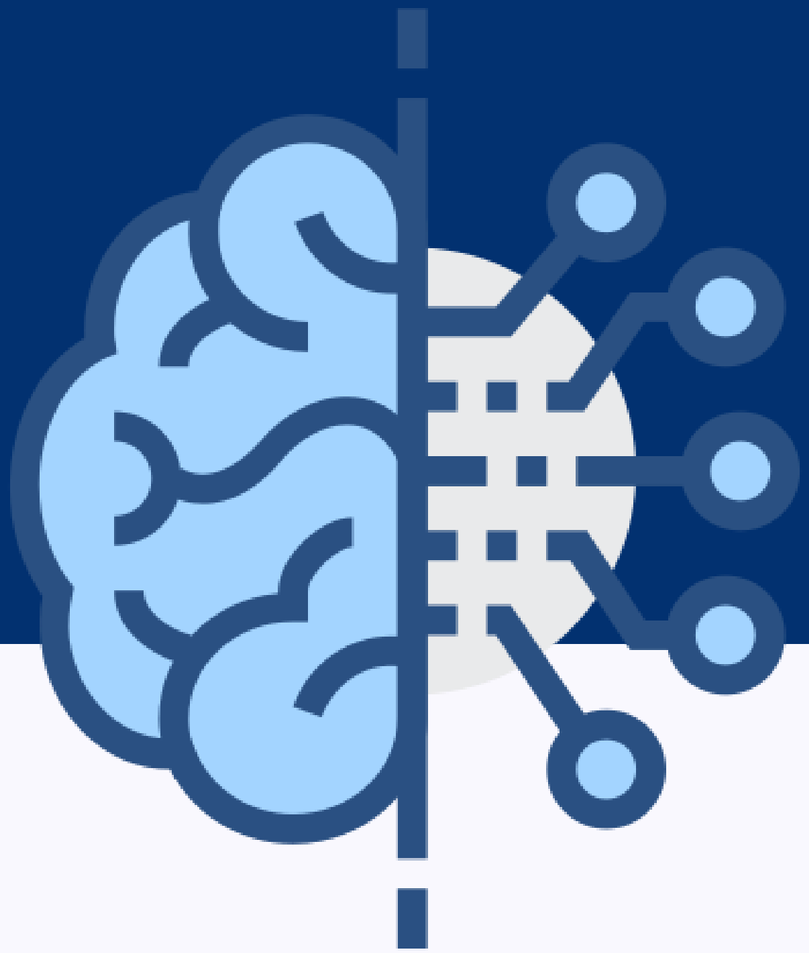
Università degli studi di Milano Bicocca

IMAGE CLASSIFICATION

Foundations of Deep Learning Project



Marco Donzella 829358, Rebecca Picarelli 834286

Table of contents

Key points of the presentation

Introduction

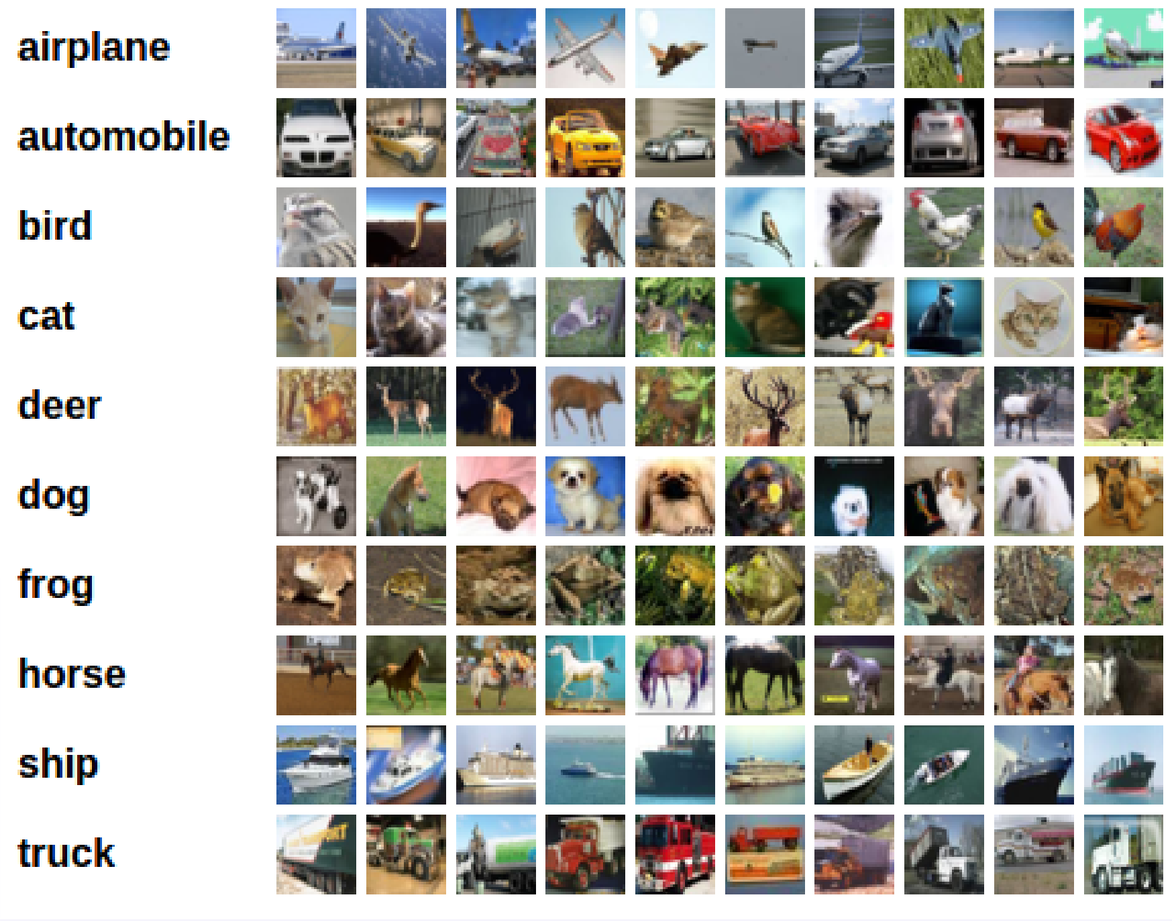
- **Aim of the project** : performing **image classification,** a subdomain of computer vision, on CIFAR-10 dataset (provided by the Canadian Institute for Advanced Research) with Python
- **Method 1** : build a neural network structure using their own knowledge (with attempts)
- **Method 2** : build a model using the transfer learning technique

The project was developed in python, using the software open source TensorFlow.

CIFAR-10

The dataset used for the project is CIFAR-10 and it's available at the following link:
http://www.cs.toronto.edu/~kriz/cifar.html

The dataset is also available in keras, because it's one the pre-downloaded dataset:
https://keras.io/api/datasets/cifar10/



• The dataset iscomposed of 60.000 32x32 colour images (RGB) divided into 10 mutually exclusive classes (each with 6000 images)
• 50k used as training set and 10k as test set

Preprocessing

- **Data normalization :** dividing by 255.0, so the new range is [0-1] and conversion to float
- **Data standardization :** for each of the 3 channels
- **Target :** transformation with one-hot encoding (10 element binary vector with 1 as the index of the class value)

Data augmentation

- Rotation range = 15
- Horizontal flipping
- shift images horizontally and vertically with range 0.1

The model

Here is the structure of the best model built (after lots of attempts).

```
model = Sequential()
model.add(keras.Input((32,32,3)))
model.add(Conv2D(32, (3, 3), activation='Mish', padding='same', kernel_initializer='he_uniform',kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(BatchNormalization())

model.add(Conv2D(32, (3, 3), activation='Mish', padding='same',kernel_initializer='he_uniform',kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='Mish', padding='same',kernel_initializer='he_uniform',kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='Mish', padding='same',kernel_initializer='he_uniform',kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3, 3), activation='Mish',  padding='same',kernel_initializer='he_uniform',kernel_regularizer=keras.regularizers.l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation='Mish',kernel_initializer='he_uniform') )
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

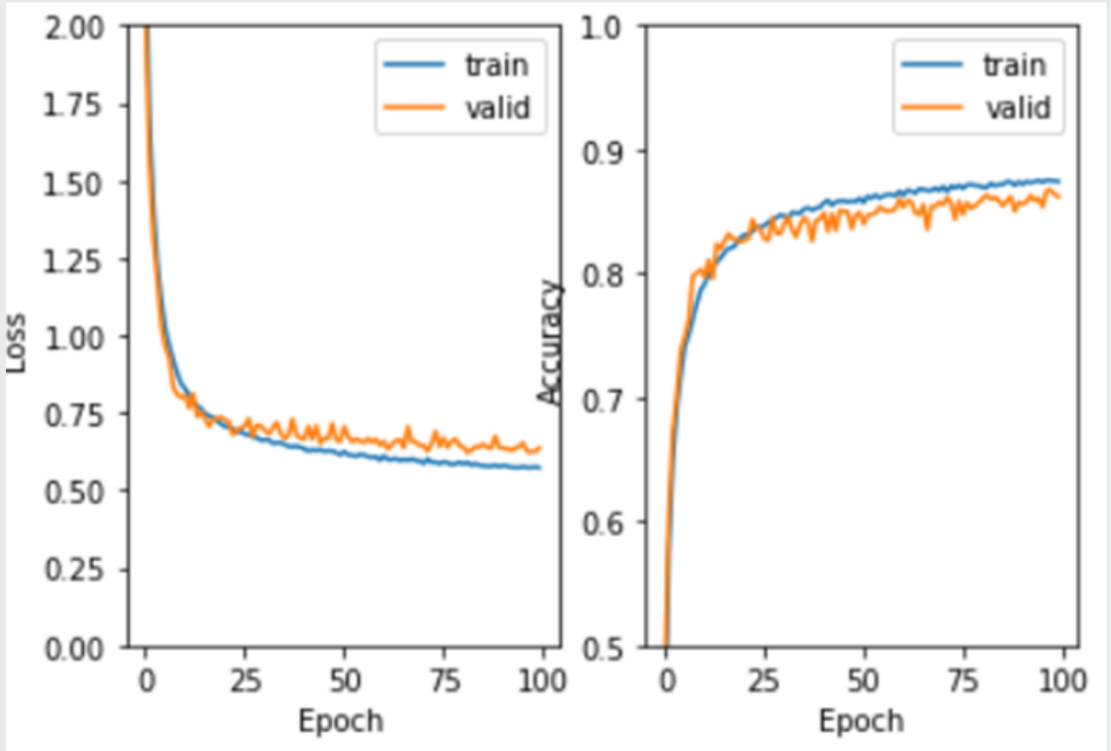
Performance

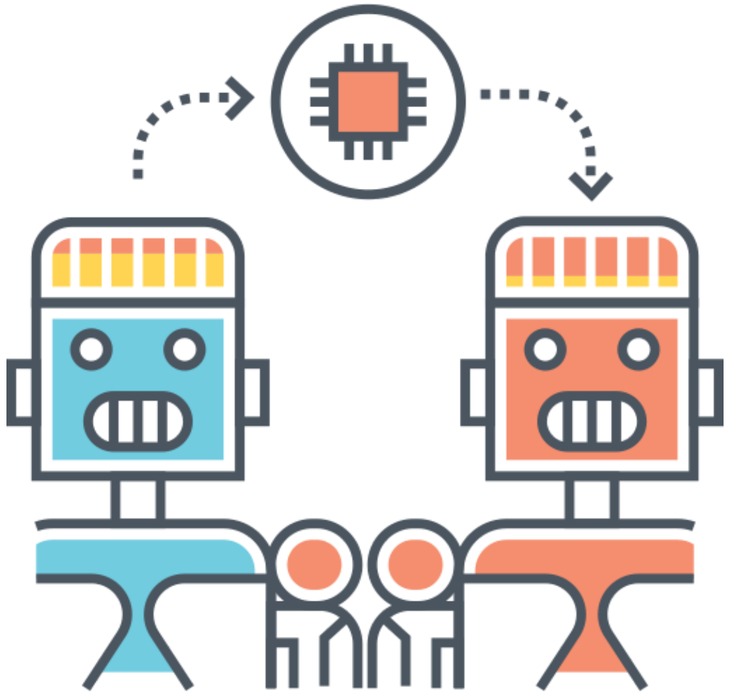
The model was trained with the following parameters:

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

```
history = model.fit(x_train, yc_train, batch_size=128,steps_per_epoch=x_train.shape[0]//128, epochs=100, verbose=1,
          validation_data=(x_test, yc_test),
          callbacks=[keras.callbacks.EarlyStopping(monitor='accuracy', patience=10)])
```

The accuracy on the test set appears to be always around 86% without overfitting. To get the output it takes around 15 minutes.





Transfer learning

Transfer learning

The steps performed to build the new model

THE BASE MODEL

The model **InceptionResNetV2** was used as base; it's one of the most famous keras applications: https://keras.io/api/applications/inceptionresnetv2/

PREPROCESSING

In the new model were implemented the following steps of preprocessing:
- image resize (from 32x32 to 299x299)
- 3 ways of data augmentation (horizontal flip, translation, rotation)
- resnetV2 preprocessing

CHANGE

The last part of the new model was defined with dense layers, batch-normalization layers, dropout layers and the final fully connected layer for the classification problem (10 labels)

The final model

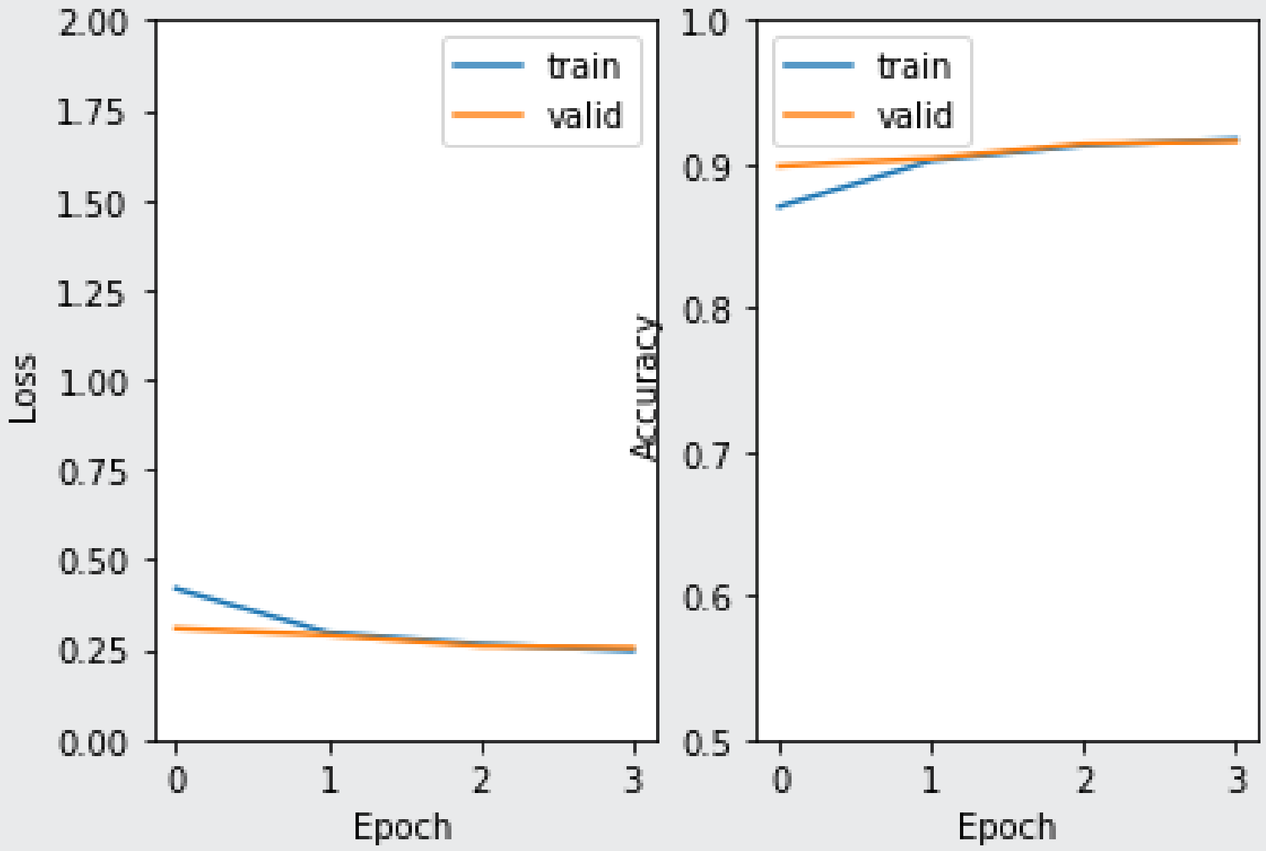
Here is the structure of the final model. The convolutional layers of the InceptionResNetV2 were freezed, only the classification layers were changed.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 32, 32, 3)]	0
lambda (Lambda)	(None, 299, 299, 3)	0
tf.math.truediv (TFOpLambda)	(None, 299, 299, 3)	0
tf.math.subtract (TFOpLambda)	(None, 299, 299, 3)	0
inception_resnet_v2 (Functional)	(None, 8, 8, 1536)	54336736
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1536)	0
dense (Dense)	(None, 512)	786944
batch_normalization_203 (BatchNormalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65664
batch_normalization_204 (BatchNormalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290

Total params: 55,193,194
Trainable params: 855,178
Non-trainable params: 54,338,016

Performance

The accuracy of the new model is improved and reaches the value of 0.92. The model is deeper and heavier to execute than the previously. The model was fitted with batch_size=256, epochs=4, optimizer=Adam, loss=categorical_crossentropy.

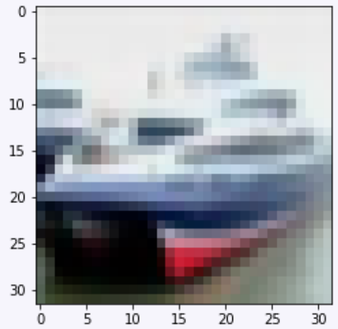



```
Classification report:
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      1000
           1       0.93      0.97      0.95      1000
           2       0.97      0.83      0.90      1000
           3       0.84      0.87      0.86      1000
           4       0.85      0.93      0.89      1000
           5       0.91      0.89      0.90      1000
           6       0.92      0.94      0.93      1000
           7       0.96      0.94      0.95      1000
           8       0.92      0.96      0.94      1000
           9       0.96      0.90      0.93      1000

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000

Confusion matrix:
[[928   4   4   1   4   0   1   2  49   7]
 [  5 967   0   2   0   1   1   1   4  19]
 [ 28   0 832  35  61   8  25   7   4   0]
 [  8   2   7 869  24  54  26   2   6   2]
 [  5   0   5  12 931   6  18  21   2   0]
 [  1   0   2  70  22 892   5   8   0   0]
 [  4   1   3  28  16   9 935   1   3   0]
 [  5   0   4  11  31   7   0 941   1   0]
 [ 21   4   1   1   1   0   2   0 964   6]
 [ 10  62   0   3   3   1   0   1  20 900]]
```

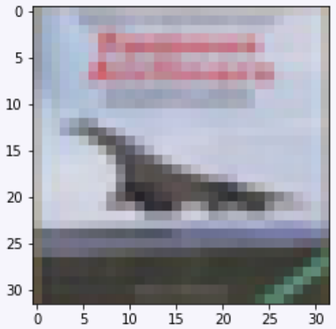
Some examples

Here is some examples of classification of the final model. The images used are from the test dataset:

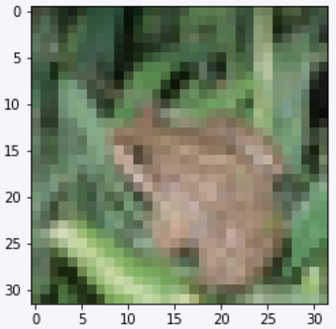
Ship



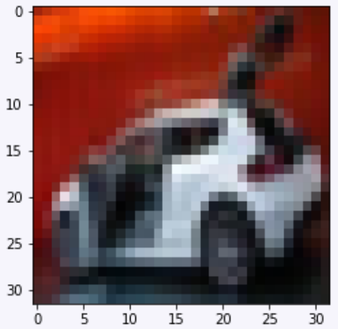
Airplane



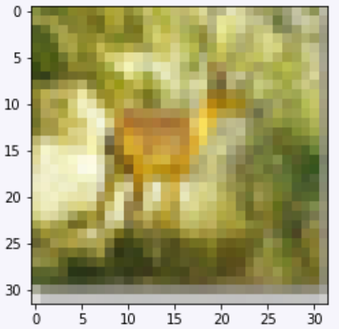
Frog



Automobile



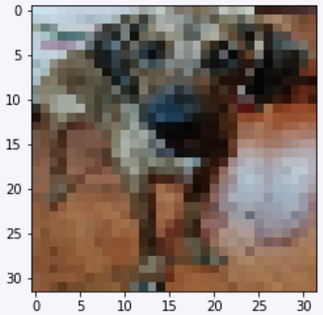
Deer



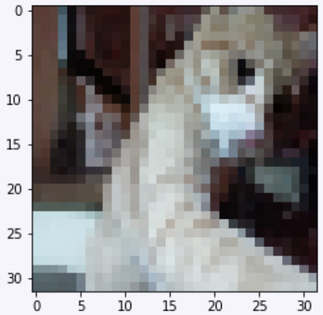
New examples

The examples shown below are new images, not present in the test set:

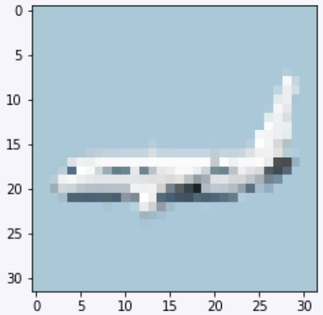
Dog



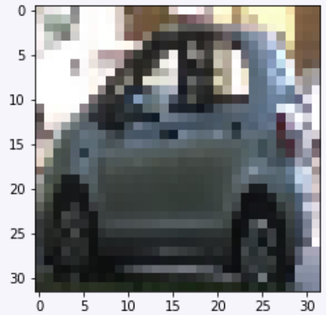
Cat



Airplane



Automobile



Conclusions

- The first part : it has been possible to find a quite simple architecture for a cnn that can lead to good results, considering the accuracy on the test set(not less than 86%), in a reasonable amount of time.

- The second part : the transfer learning was useful, in fact the accuarcy of the model is improved and increased from 0.86 to 0.92. The final model is more complex and heavier than the first one. As appropriate, the first model (smaller and faster) could be more suitable to use.

Future developements

01 Try other techniques in the data augmentation field to see if results improve

02 Estimate the best hyperparameters with some tuning steps to improve the net architecture

03 Use more powerful processors and increase the number of epochs

04 Try some models from the VGG family to improve the results

05 Develope the same procedures for the CIFAR-100 dataset and compare the results

Thanks for the attention