Session 6
Discussion Forum 3
Convolutional Networks for Images

Organization of DISCUSION FORUMs

The discussion forum are exercises/challenges that will test practically the information shared in On-line classes. You must write a program and execute it in a notebook format. At the end of the week you must submit this notebook. The submission deadline date will be Saturday before the on-line class.

The notebook must contain the responses to some questions. To respond them, use a cell in the notebook with the answers.

During the week (Monday to Friday) you can post in the forum sharing your questions, your thoughts and your accomplishments to the group. I will try to answer all your posts trying to guide you to solve the exercise and any other question that you may raise.

Learning Objectives

The objective of this Exercise is to create a convolutional neural network to classify a large dataset of photos of Cats and Dogs. The algorithm will be a binary classification of photos that will tell when a photo is of a cat and when a photo is of a dog.

You will learn the implications of working with a large set of photos or working with a small one, the impact of modifying the convolutional network architecture and the impact of image augmentation.

Files for this Practice

```
060_CNN_Cats_and_Dogs_Keras-Baseline.ipynb
040_CNNCIFAR_KERAS-Baseline.ipynb
cat.jpg
```

Some notes about Deep Learning environments

This Practice is to refresh how to create ANN to solve Deep Learning problems using KERAS, and to fine tune your environment to accomodate the Keras-Tensorflow packages and make sure they work together with yoour gymnasium ones.

There are two major tools for neural networks. Tensorflow-Keras and Pytorch. We will focus on KERAS in this course, however both tools are thriving and you can find many application in one or the other.

PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab (FAIR). It has gained immense popularity in the machine learning community due to its dynamic computational graph, ease of use, and flexibility. PyTorch provides a comprehensive platform for building and training deep learning models, making it an excellent choice for both research and production. Keras has been developed by Google.

There are several resources for Deep Learning, I can recommend [Tra19], for Pytorch you can find a good introduction in [SAV20]. If you are interested in the best introduction to Neural Networks / Deep Learning look at this book, from François Chollet, A senior Google engineer and the author of Keras [Cho17], be sure you are reading the 2nd edition which is reviewed and more recent.

1 Flattening and reshaping matrixes

Flattening an array consists of changing its shape. There is a visual representation of this operation in Figure 1,

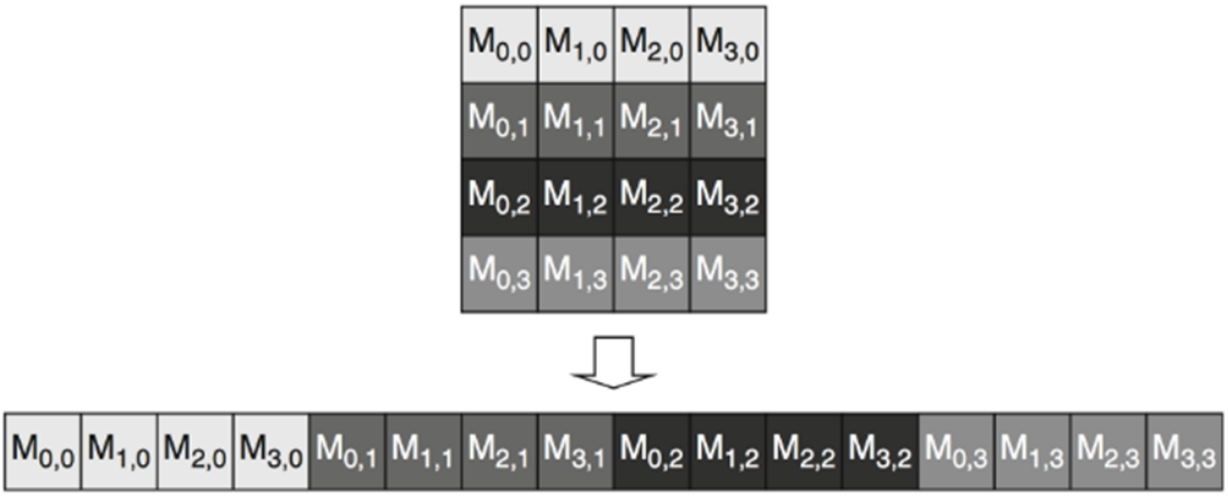


Figure 1: Flatten - Reshape operation in a 2D matrix

Numpy arrays can be reshaped in many ways using the .reshape() function. The Flattening operation is important because when we transition from the convolutional operations we usually transform the data into a one dimensional tensor.

Remember, in the convolutional operation the input shape of the data is (height, width, 3) being 3 the three colors RedGreenBlue in the image representation

The convolutional operation expects these 3 channel tensor that is used as input in the Deep Learning CNN networks.

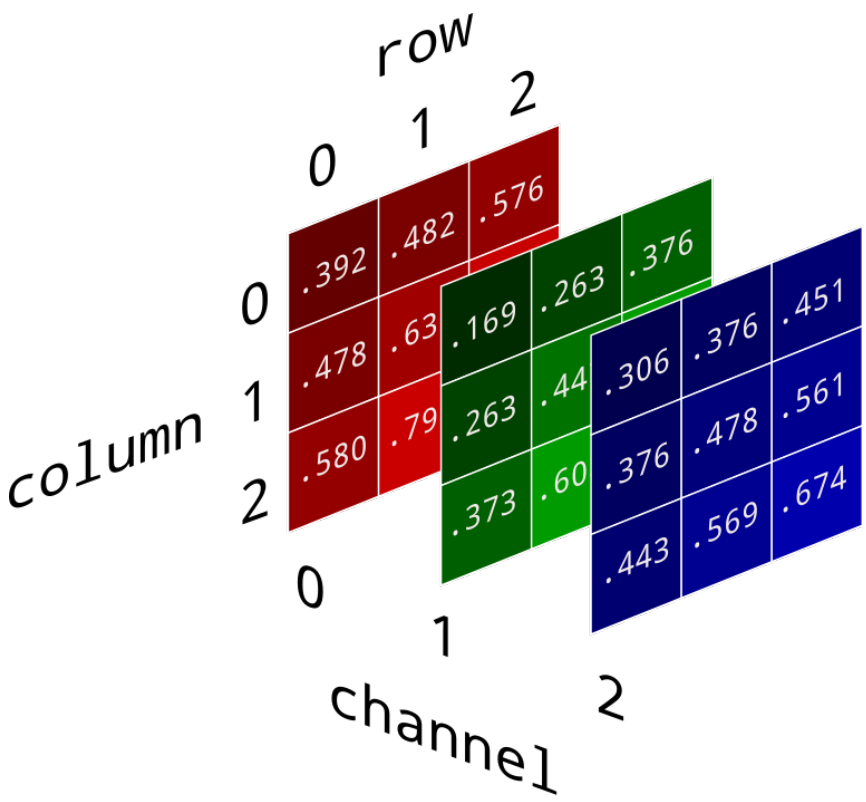


Figure 2: How the images are represented in Deep Learning

2 Back to the CIFAR example

In the CIFAR10 example with MLP we obtained results on the 50s of accuracy. We tweaked the program in any possible dimension and we did not obtain any major breakthrough. Do we have any other way to extract information from those tiny images?. The answer is Yes. Let's apply a convolutional network to this problem.

We need to do the following changes in the last week program.

Start with the CNN Baseline program

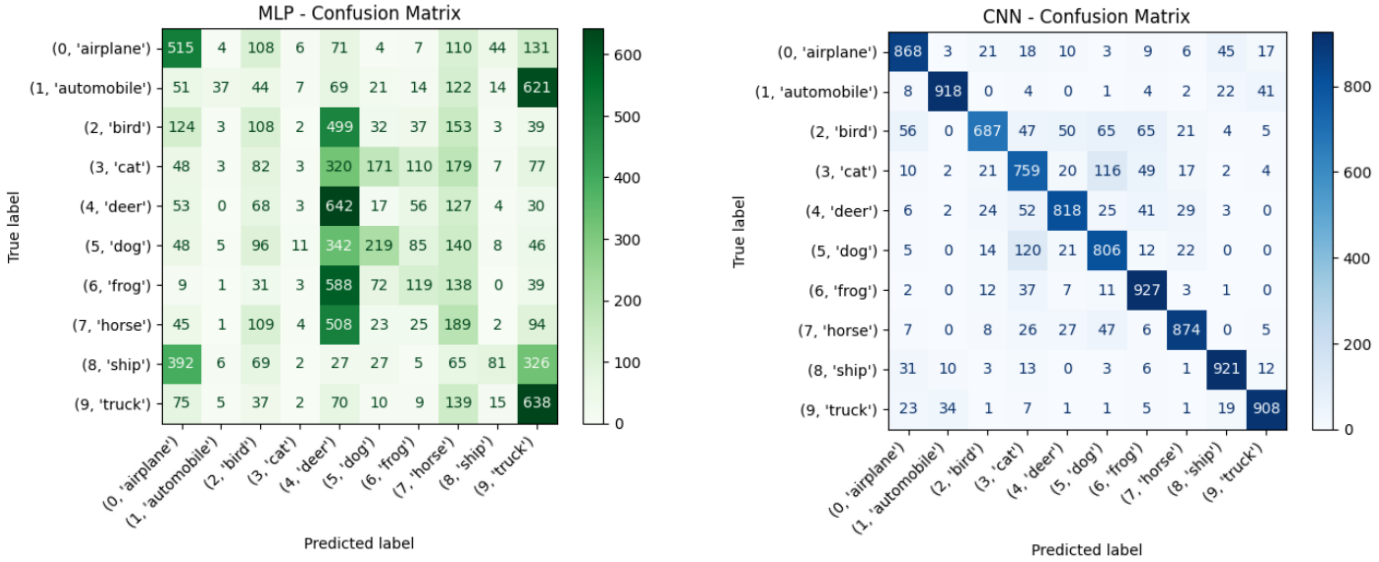
```
040_CNN_CIFAR_KERAS-Baseline.ipynb
```

1. We need to change the input shapes. When feeding MLP's we usually do it by using a flat array, in this case the input shape of the array for the MLP is (3072) in this array we have 'Flattened' the three matrixes $3x(32x32)$. When using convolutional networks we maintain the original structure, in this case (32, 32, 3) which are the 3 image matrixes. As we said in class, shape management is a key component when using Deep Learning. You must have a clear view on what is the input shape of the Network Architecture, and how the shapes evolution after each operation.
2. Instead of an MLP we need to create a Convolutional network. In the baseline program you have just one block layer. As we shared in the class, convolutional networks structure is built using 'blocks'

or repeatable sets of layers. In this case I recommend you to use 3 blocks, similar to the ones you already have in the Baseline Program.

3. Remember a convolutional architecture always has the convolutional layers (extract features) and the classification MLP (from the features determines the class). Do not touch the MLP (Dense) layers

With only one block you will get up to 70%, however, if you use 3/4 blocks you will get very close to 90%



(a) CIFAR10 MLP Confussion Matrix

(b) CIFAR10 CNN Confussion Matrix

Figure 3: Comparison of two Confussion Matrixes CNN and MLP

3 The Cats and Dogs Dataset

The Cats and Dogs Dataset is a popular dataset commonly used in machine learning and computer vision tasks, particularly for image classification problems. It was featured in a Kaggle competition and serves as an excellent resource for beginners and experts to explore binary image classification and deep learning techniques.

originates from Microsoft Research and is part of their larger Microsoft Research Data Science Machine Learning Datasets. It was initially used for benchmarking computer vision tasks and is publicly available on platforms like Kaggle for educational and research purposes.

The dataset is derived from the Asirra (Animal Species Image Recognition for Restricting Access) project by Microsoft Research. This project aimed to explore CAPTCHA systems that use image classification to distinguish between cats and dogs.

- **Number of Samples:** The dataset contains over 25,000 training iumages
- **Image Dimensions:** Images are provided in standard formats, such as .jpg or .png. The resolution

of the images varies, with most being medium to high resolution, but resizing is often required for model training. They have different sizes and resolutions and labeled as cat or dog

- **Classes:** Two classes cats and dogs, useful for binary classifications

Example Images

Figure 4 shows examples of cats and dogs from the dataset.

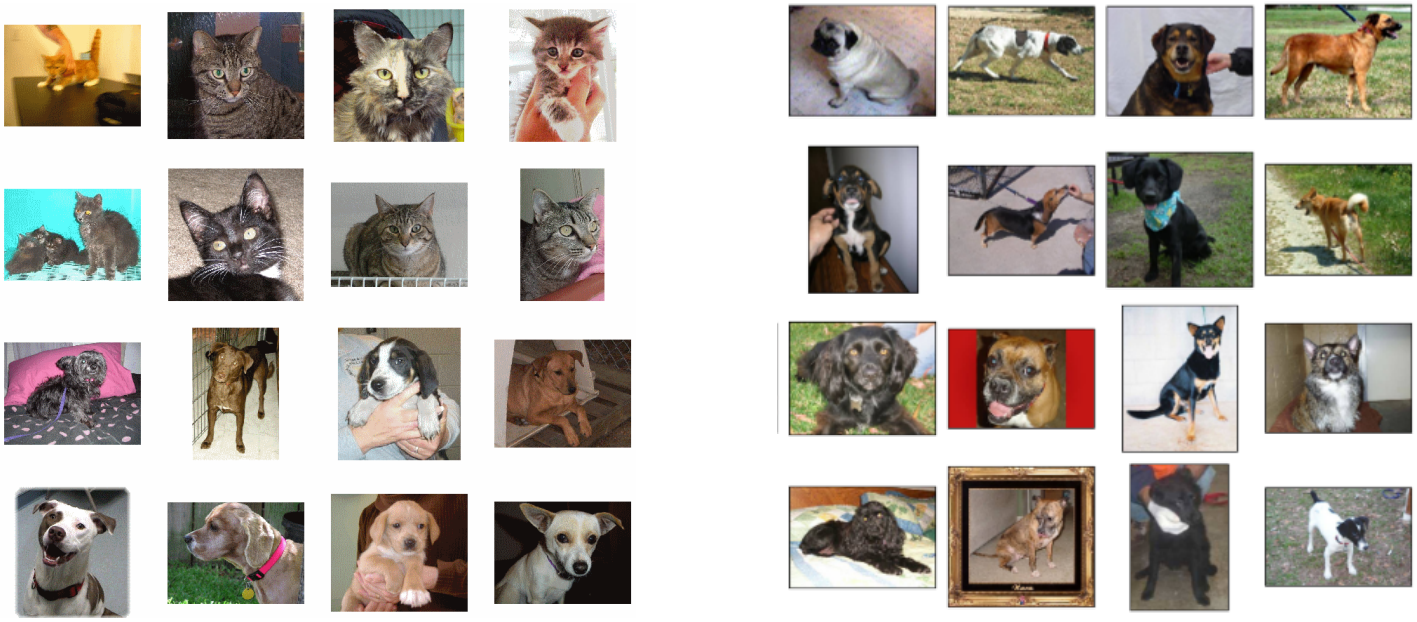


Figure 4: Images from the Cats and Dogs Dataset

Background on Keras

Keras is a high-level neural network library that was first developed by François Chollet in 2015. It was initially designed to serve as a user-friendly interface for building deep learning models, sitting on top of lower-level machine learning frameworks like TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). Over time, it became tightly integrated into TensorFlow and is now its official high-level API.

Keras is known for its simplicity and modularity, making it ideal for beginners and experts alike. It supports building neural networks with layers, offering flexibility to design both simple and complex architectures. Key features include support for convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid models.

Keras provides built-in tools for data preprocessing, visualization, and evaluation. It also supports distributed training and is compatible with CPUs, GPUs, and TPUs. With extensive community support and a wealth of pre-trained models, Keras is a go-to library for deep learning tasks in academia and industry.

(A good book to use as an introductory book to deep learning is this one [Tra19], but to follow the examples in pytorch I recomend this one [SAV20], which has the examples in torch)

4 Developing an Image Binary Classifier for the Cats and Dogs Dataset

To develop a deep learning image classification algorithm we need a set of labelled images, in this case we will use the Cats and Dogs dataset which contains thousands of images of cats and dogs.

The use of images can be a bit complex as there are many supporting functions in Keras and in other packages like PIL that simplify the image management. However we need to learn those supporting functions to be efficient. Like the saying, *"When in Rome, do as the Romans do"* and following it we developed the program using Keras supporting functions.

The basic program creates a set of directories named `PetImages` with two subdirectories `Cat` and `Dog`. It reads the database from Microsoft and loads it in those working areas.

The program works quite well in a laptop, but we prepared it in this way to make possible to work in Google COLAB without any issues. You may speed things up by downloading the file once and then working directly with the files in your computer.

Start with the `06_CNN_Cats_and_Dogs-Baseline`

After the creation of the subdirectories, the data is prepared using a supporting function from keras named `Image_dataset_from_directory`. you can check the documentation of this function, but basically transforms the images in the dataset into a stream that can feed the network. Basically we are using the tensorflow.data API. You can see the documentation of this API here:

```
https://www.tensorflow.org/guide/data
```

However we don't need to research it very well as the simplest use of it is already implemented in both programs.

5 Network Architecture

The Network Architecture used is a convolutional network with several layers.

Usually a convolutional architecture has two parts. The convolutions (feature extraction) and the MLP (classification). The convolutions are applied by blocks and many times. Here you have an example with one block. Try to replicate 2/3/4 blocks and see what happens.

```
# Images are 150x150x3: 3 for the three color channels: R, G, and B
img_input = layers.Input(shape=(150, 150, 3))

# First convolution block extracts 16 filters that are 3x3
# Convolution is followed by max-pooling layer with a 2x2 window
x = layers.Conv2D(16, 3, activation='relu')(img_input)
```

```
x = layers.MaxPooling2D(2)(x)

.............
.............

# Flatten feature map to a 1-dim tensor
x = layers.Flatten()(x)

# Create the MLP Layer
x = layers.Dense(512, activation='relu')(x)

x = layers.Dropout(0.5)(x)
# output layer single node/sigmoid activation - binary classification
output = layers.Dense(1, activation='sigmoid')(x)

# Configure and compile the model
model = Model(img_input, output)
model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['acc'])
model.summary()
```

Try to get as close as you can to 90% in this exercise.

6 What should be done in the Assignment

You must try the Cats and Dogs code and see if you can get a result close to 90%. Try the CIFAR code as well and see if you get it to 80% but you don't need to include the code in the assignment.

6.1 Try Alternative structures of the Convolutional Network

Increase the size of the convolutional network by adding layers and adding filters to each layer.

The new layers may have more filters (this is the first number in the Conv2D layer, you can see it in the documentation below.

In general, the filters grow in a pyramidal structure. Use filters 3x3 but you can increase them as well if you want. Basically create blocks and repeat them. An idea will be to try to replicate a VGG16 architecture (you have it in the class material). The VGG16 structure is well tested and works very well for images.

You can get inspiration from other examples in the literature (in internet basically). However, think that getting over 90% is very difficult so don't overcomplicate your experimentation.

Finally a recommendation. Moving images around is computationally expensive. I am almost sure that you need to use COLAB or if you are using your laptop CPU your computing times will be long for some of the exercises. Now you will understand why the NVIDIA valuation is so high (as the computing required for Deep Learning grows by the day).

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 150, 3)	0
conv2d (Conv2D)	(None, 148, 148, 16)	448
max_pooling2d (MaxPooling2D)	(None, 74, 74, 16)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
conv2d_2 (Conv2D)	(None, 34, 34, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 64)	0
flatten (Flatten)	(None, 18496)	0
dense (Dense)	(None, 512)	9,470,464
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 9,494,561 (36.22 MB)

Trainable params: 9,494,561 (36.22 MB)

Non-trainable params: 0 (0.00 B)

Figure 5: Structure of convolutional network using model.summary()

```
keras.layers.Conv2D(
    filters,
    kernel_size,
    padding="valid",
    activations='relu
)
```

Let's see in your exercise to answer the question What is your best Accuracy?

Summarize your experimentation and explain how you ended up in the final best performing architecture

7 Questions to Answer

1. How do you explain the changes in the accuracy by increasing the architecture size ?
2. What is better more data or better algorithm?
3. How did you get to your final architecture?

8 Deliverables and submission

Submit the following files with content

- A Notebook with the CNN program with the answers to the questions in the last cell

Epilogue

And Last but not least. If you are interested in this field you definitively must read the book The Alignment Problem, there you have a chapter explaining how the first Convolutional network was applied to images by Hinton and his student Alex Krizhevsky.[Chr20].

If you want to read the original paper you can access it here [KSH12]. If you look at this article you will discover a name in the authors list. Ilya Sutskever, possibly you know he is in the board of OpeanAI. (Hinton is receiving the Nobel 2024 of Physics any time soon). This article has been cited more than 100.000 times.

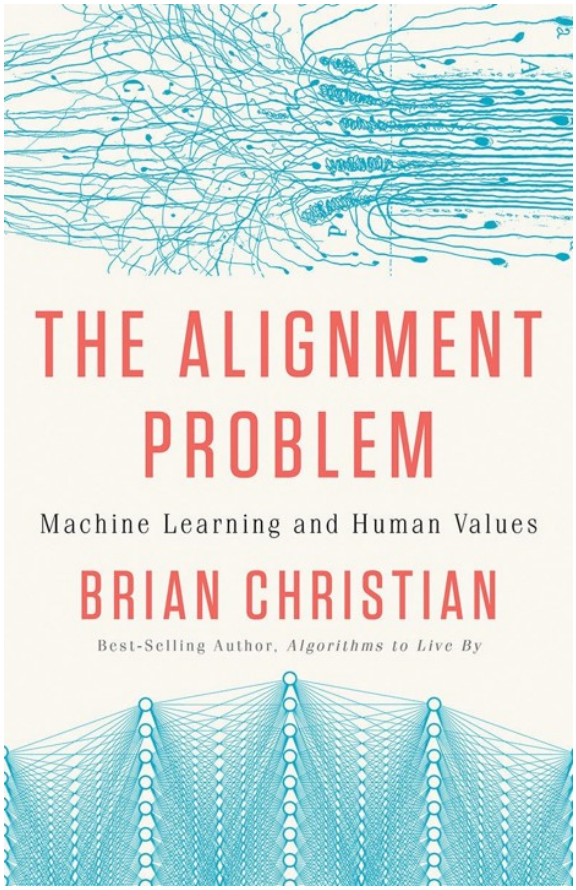


Figure 6: A "Must Read" book

References

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[Cho17] François Chollet. *Deep Learning with Python*. 2nd ed. Manning, Nov. 2017. ISBN: 9781617294433.

[Tra19] Andrew Trask. *Grokking Deep Learning*. 1st. USA: Manning Publications Co., 2019. ISBN: 1617293709.

[Chr20] B. Christian. *The Alignment Problem: Machine Learning and Human Values*. WW Norton, 2020. ISBN: 9780393635829.

[SAV20] Eli Stevens, Luca Antiga, and Thomas Viehmann. *Deep Learning with Pytorch*. Manning, 2020. ISBN: 9781633438859.