# CS4487 MACHINE LEARNING PROJECT

## Image classification CIFAR-10

**Veljko Kovac (55173123)**


香港城市大學
City University of Hong Kong

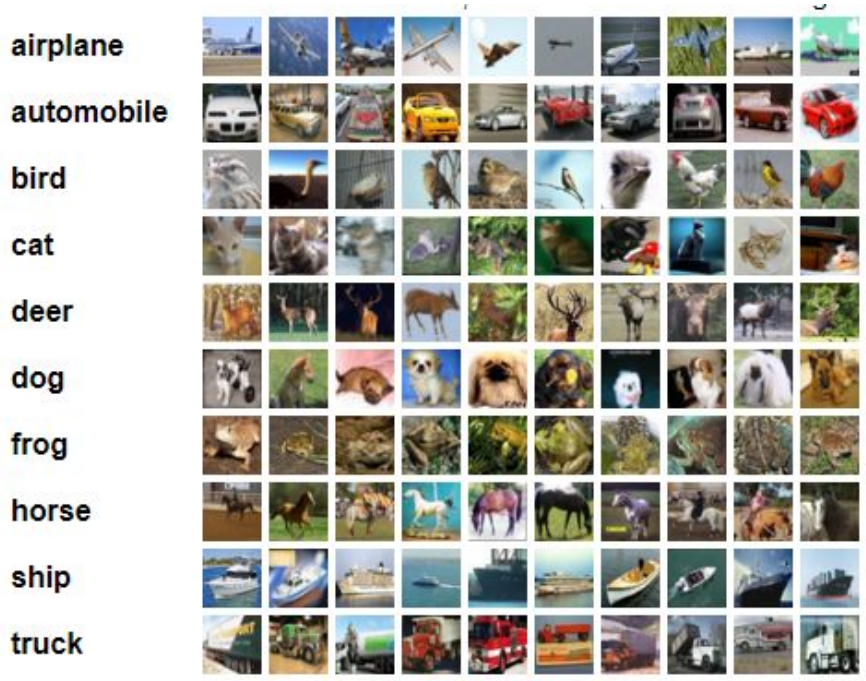
# Table of Contents



# Introduction

**CIFAR-10 DATASET:**

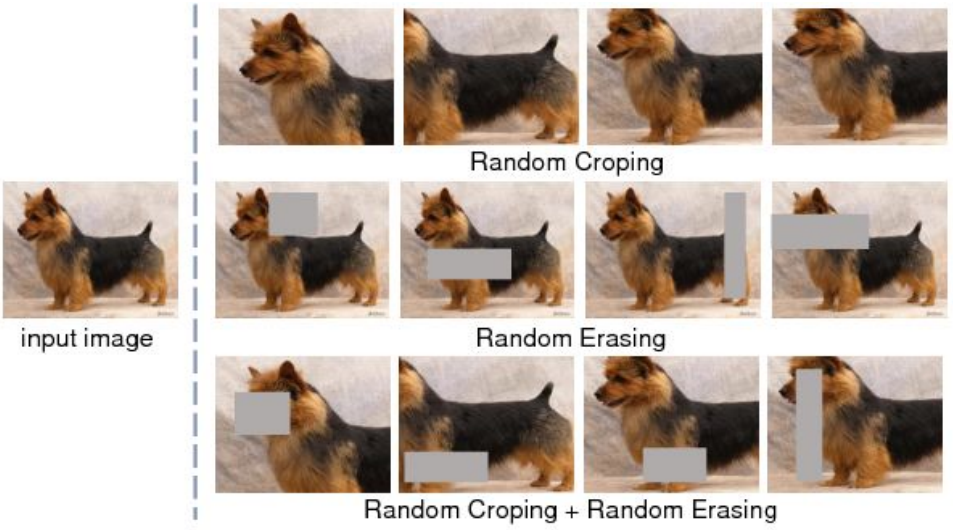
For this project, we had to classify the CIFAR-10 Dataset into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. CIFAR-10 is a famous dataset where many famous algorithms for image classification were developed. Its size is 60000 (6000 of each class) and is divided into 2 sets, the training with size 50000 and the testing of 10000. For this project, however, we were asked to submit a CSV file with our predictions of an extra given test data with size 12000. All pictures are in RGB with 32x32 pixels.



Ref. [1]

# Pre-Processing Techniques

CIFAR-10 is an already clean dataset that does not need any further cleaning. However, some steps should be done in order to get better model performance. Firstly, because deep learning needs a lot of data, a good idea to increase them is by using Data Augmentation. Data Augmentation is a process of creating new data points by manipulating the already existing training data. This gives us the chance to increase our train set without actually collecting more. Some techniques that we used for our case, is Random Crop and Random Horizontal Flip. It is easily understandable that random crop modifies a given image by randomly cropping it while a random horizontal flip by flipping it. Another pre-process technique that we used was Normalisation. We transformed all our data to the same scale within [0,1] by subtracting from each value the mean and dividing it by the standard deviation. Lastly, because we have chosen to use Pytorch for our project, we had to transform all the pre-processed data to tensors. For our project we chose to use Convolutional Neural Networks, since is the most efficient technique for image classification.




Ref. [2]

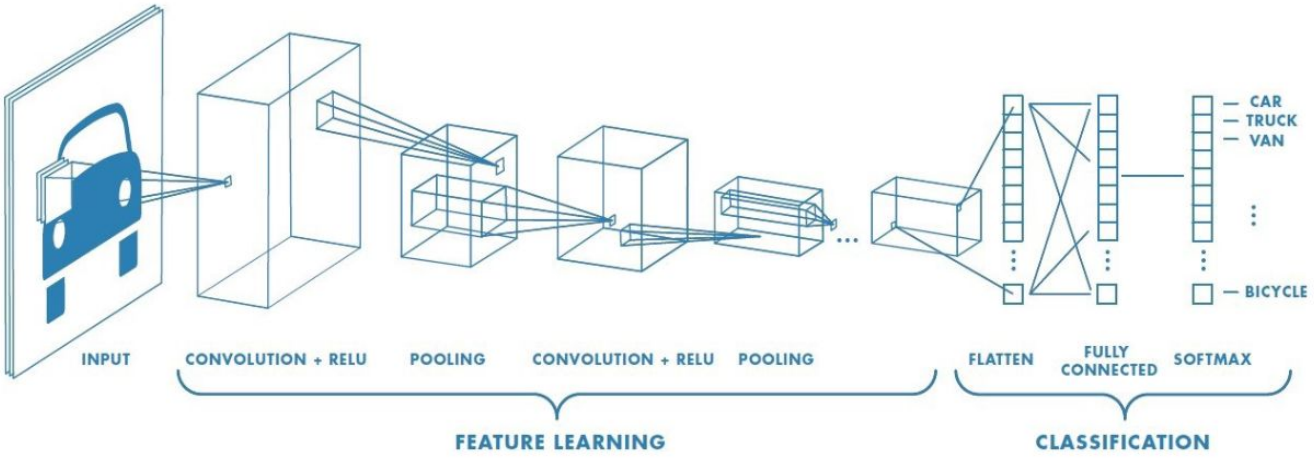
# Convolutional Neural Networks (CNNs)

One of the most important deep learning algorithms that were developed for image classification is the use of Convolutional Neural Networks, also well known as CNNs. CNNs emerged from the study of the brain’s visual cortex in the 1980s and has been

used apart from image classification in Voice Recognition, Natural Language Processing, and many other machine learning tasks.

Convolution networks are able to capture in a really good level the spatial and temporal dependencies in an image through the application of relevant filters. What this means, basically, is that by reducing the number of parameters, and reusing the weights. it understands the image sophistication much better than other deep learning algorithms.

What the convolutional network does, is to try to reduce the dimension and size of the image data without losing important features of the image. This is achieved by the use of multiple convolution and pool layers. It is worth mentioning that the more layers you have, the more memory you need.




Ref. [3]

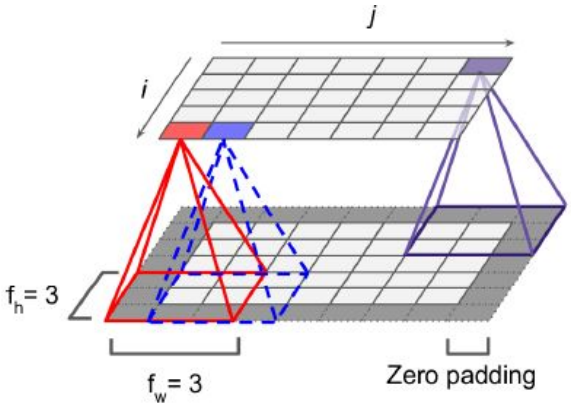
**Convolutional Layer**

It is really important to understand the way convolutional layers work. As the primary goal is to find any potential patterns in an image, every single layer will find something important in an image. By saying that, it is important to understand that the deeper the network the more complex patterns will the Convolutional Layers will find. What I mean is that the first convolutional layer will start detecting the more simple features, like edges, corners, colors etc and as it goes deeper the network will give us a better wholesome understanding of the image.

The way it works is really impressive. In the first convolution layer, neurons are not connected to every pixel of the image, but only to pixels in their receptive fields. As it goes on the second and third convolution layer, the neurons are connected only to

neurons located within a small rectangle in the previous layer. With this method, the CNNs are able to concentrate on small low-level features in the beginning, and as the network proceeds deeper, they are able to assemble them into larger higher-level features which are ideal for image recognition.

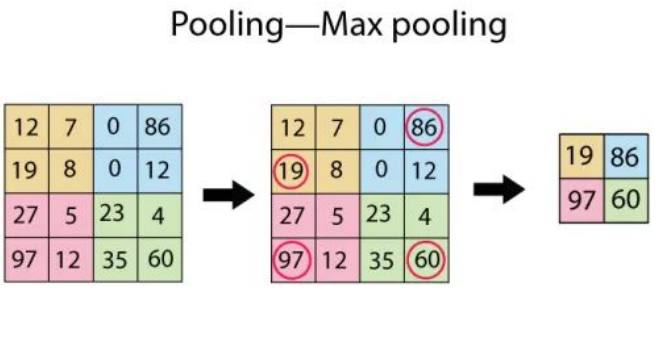
More technically, after every convolution layer, the dimension will be decreased because we might lose some pixels on the outside. In order for the output of a layer to have the same height and width as the previous one, we introduce the concept of padding. By implementing zero-padding zeros will be added around the new inputs. The number of rows and columns traversed by slides are referred to as strides and in our project was equal to one. Worth mentioning that explanations and definitions were taken from the book written in reference [4].




Ref. [4]

**Max Pooling Layer**

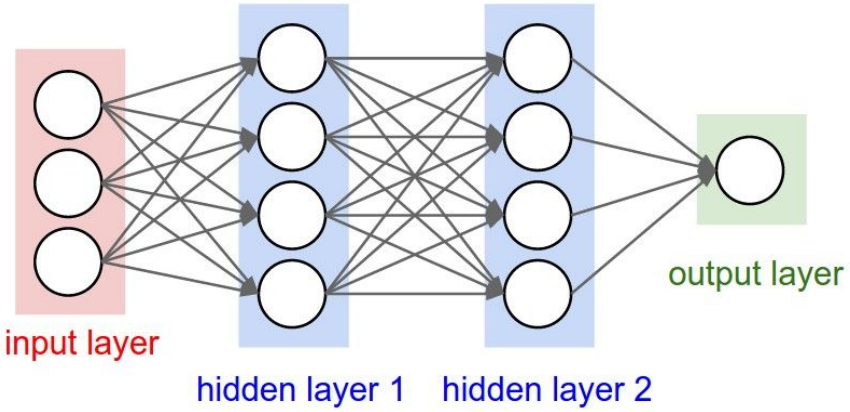
However, convolution layers are not the only layers that are used in CNN. Another important layer that must be comprehended is Max Pooling Layer. Usually, after one or two convolution layers it is common to use and one max-pool layer in order to decrease the spatial size of the convolved features. It is basically extracting the most dominant features by giving us the maximum value from the portion of the image covered by the kernel.




Ref. [5]

**Fully Connected Layer:**

After using several convolutional and max-pool layers, there is a step of flattening the image into a column vector so it can get through the fully-connected layer. Fully connected layers are one of the best ways to learn non-linear combinations of high-level features, and that is the reason we are using them. The flattened output is fed to our neural network, where forward and backward propagation takes place over a series of epochs in order to find the possible weights that will make the model able to distinguish how important is a feature. The last step is always the classification in categories, which is done by using the Softmax Layer. The output of each class will be limited to [0,1] which automatically becomes the probability of belonging to that specific class. It is easy to understand, that the final output of the model will be the class with a higher probability.
There are many different kinds of CNN architectures where each of them will have a specific combination of layers or will be adding something unique. Different CNN architectures will be discussed further below.




Ref. [6]

# Types of Architectures

Computer Vision and Deep Learning literature is flooded with papers applying CNNs to image classification [7,8,9,10,11,12]. There are a few very famous architectures with record performance on benchmark datasets: Imagenet, Stanford Cars, CIFAR-10, CIFAR-100, etc. Among these architectures are some that got a disproportionate amount of attention from the Machine Learning community (and rightfully so given their merit). We will briefly describe three famous architectures in the next few paragraphs.

1. **DenseNet** [10]: Since the discovery of Neural Networks, the search for optimal architectures has always been a problem of great interest. Previous work on ReseNets, Highwaynets, and FractalNets suggests that the search to improve the accuracy of convolutional networks has been largely reliant on shorter connections between the input and output layers. In the same spirit, and to ensure full connectivity, DenseNet connects ***all*** layers with each other. Consequently, this introduces (L+1)(L)/2 connections in an L-layer network. To put it another way, feature maps of all preceding layers are used as input in any layer. Using such an architecture has some compelling advantages as the authors claim: alleviation of vanishing gradients, strengthening of feature propagation, feature reuse, and reduction in the number of parameters.

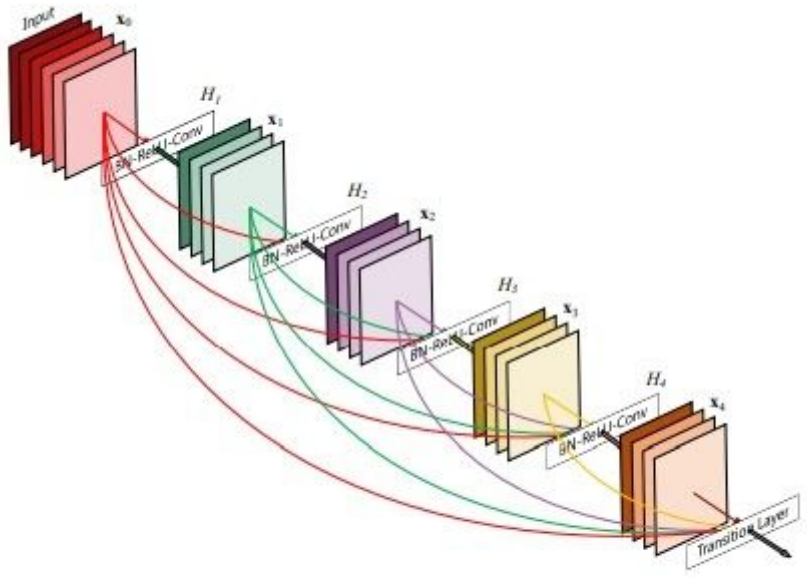
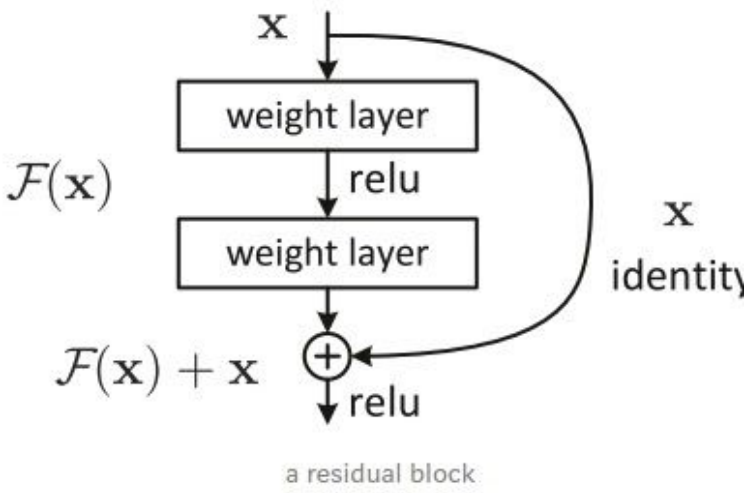



**Figure 1:** A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Ref.7

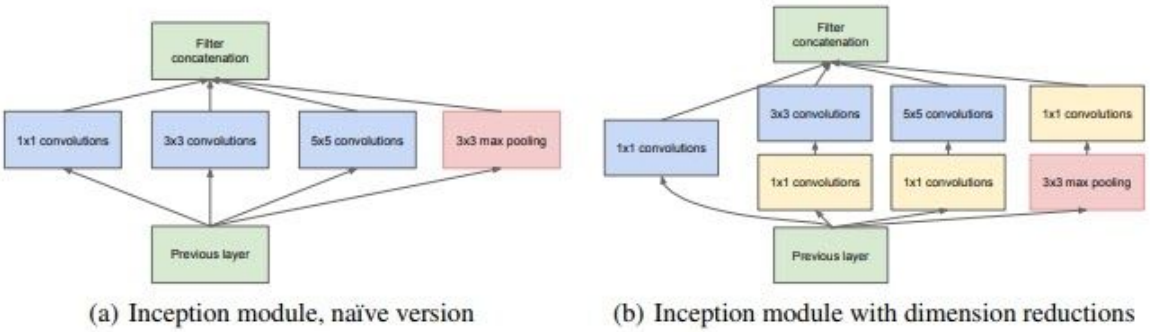
2. **ResNet** [11]: ResNets, as they are called, use the so-called ‘residual learning’ paradigm. With the advent of AlexNet[13], researchers in machine learning started chasing for deeper and deeper networks. While AlexNet only had 5 convolutional layers, the long-standing problem of vanishing gradients was there, i.e as you propagate the gradients back in layers, the got infinitesimally small. Thus, the residual learning framework introduces the idea of “identity shortcut connections” that skips layers in between. However, ResNet was not the first one to make use of shortcut connections. Highway Network [14] introduced the idea of Gated Shortcut Connections, and ResNet can even be (technically) conceived of a subclass of Highway Networks.




Ref.8

3. **GoogLeNet** [12]: Inspired from the meme, “We Need to go Deeper”, GoogLeNet uses the word Inception for its architectural design and stacks 27 layers into the original “Going deeper with convolutions” paper[12]. GoogLeNet is a CNN with an added “inception layer”. As the authors of the paper said:

   “*(Inception Layer) is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with*




(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

*their output filter banks concatenated into a single output vector forming the input of the next stage*.”

# “ Ian BadFellows’ ” Implementation

**Architectures**: We have implemented all three architectures described above and have reported the corresponding accuracies on CIFAR-10 dataset with 10,000 images (in addition to the ones uploaded on Kaggle).

| **Architecture** | **epochs** | **Accuracy** |
|---|---|---|
| DenseNet121 | 350 | 90.610 |
| GoogLeNet | 350 | 90.540 |
| ResNet18 | 350 | 90.810 |

The accuracies on the standard CIFAR-10 test was found to be much better than on the 12,000 images on Kaggle.

**Training**: The models were trained on Google Colab’s free GPUs. The learning rate was manually altered throughout the training process as was deemed appropriate. To make this alteration possible during training, we had to abort and resume training that was achieved by a technique called ‘checkpointing’. Since Google Colab requires only 12 hours of continuous training, we needed to make sure if our model stopped at some point, we would not lose the training weights although our model was trained in about 5-6 hrs on Colab. Checkpointing eases the burden on memory as well. And the testing was done on a CPU after exporting the trained weights from Colab. However, if the results are to be reproduced, the authors would recommend tuning the learning rate according to the following heuristic:

| **epoch** | **learning rate** |
|---|---|
| 0-120 | 0.1 |
| 120-250 | 0.01 |
| 250-350 | 0.001 |

**Code Organization**:
The code we have submitted is very reproducible that anyone should be able to run executing ‘python main.py’ in the project folder. There are, however, additional notes to consider about the organization of the project if one needs to reproduce different architectures, find the model weights, modify the .ipynb notebooks (note that the notebooks need to be run on Google Colab):

1. The above accuracies can be found in denset.ipynb, googlenet.ipynb and resnet.ipynb in the folder, where the method of checkpointing is also demonstrated using the --resume and --lr flag.
2. All the architectures implemented can be found in the ‘models’ folder
3. The main.py file should be run to execute the code. The default model is ResNet18. If you wish to change the models, comment/uncomment lines 52-54.
4. During the training process, the parameters of the model corresponding to the best accuracy are saved as ‘ckpt.pth’ file in the checkpoint folder, which can later be loaded into state_dict to evaluate the model.

# References


1. https://www.cs.toronto.edu/~kriz/cifar.html
2. https://www.semanticscholar.org/paper/Random-Erasing-Data-Augmentation-Zhong-Zheng/2788a2461ed0067e2f7aaa63c449a24a237ec341
3. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
4. Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow, by Aurelien Geron
5. https://principlesofdeeplearning.com/index.php/2018/08/27/is-pooling-dead-in-convolutional-networks/
6. http://inspirehep.net/record/1672081/plots
7. https://arxiv.org/pdf/1409.1556.pdf
8. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
9. https://arxiv.org/pdf/1409.0575.pdf
10. https://arxiv.org/pdf/1608.06993.pdf
11. https://arxiv.org/pdf/1512.03385.pdf
12. https://arxiv.org/pdf/1409.4842.pdf
13. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
14. https://arxiv.org/abs/1505.00387