Neural Networks: Convolutional Neural Networks

Author: Cristinel Popescu

The algorithm general presentation

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly used to analyzing, computer vision, AI. They are known as shift invariant or space invariant artificial neural networks (SIANN), based on their weights architecture plus translation invariance characteristics. They have applications in image and video recognition, choosing systems, image classification, medical image analysis, natural language processing and voice assistance for example Siri from Apple Computers or Alexa from Amazon, brain to computer interfaces, stock market prediction [1].

CNNs are regular versions of multilayer perceptrons. Multi layer perceptrons usually mean fully connected networks, more precisely, each neuron in one layer is connected to all neurons in the next layer. The "fully connectedness" of these networks makes them flat to overfitting data. Some typical ways of regularization include adding a form of magnitude measurement of weights to the loss function. Convolutional neural networks take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble them into more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, ConvNets are on the lower extreme [1].

Convolutional networks were inspired by biological brain processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual patch [1].

CNNs use relatively little pre processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand engineered by machine learning. This independence from prior knowledge and human effort in features design is a big advantage [1].

As classic architecture a convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNNs typically consist of a series of convolutional layers that convolve with a multiplication. The activation function is commonly named a ReLU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution [1].

When programming a CNNs, the input is a tensor with shape like the following example: (number of images) x (image height) x (image width) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels). A convolutional layer within a neural network should have the following attributes: [1]

- Convolutional kernels defined by a width and height (named commonly hyper parameters).

- The number of input channels and output channels (hyper parameters).

The depth of the conv filter (more precisely the input channels) must be equal to the number channels (depth) of the input feature map.

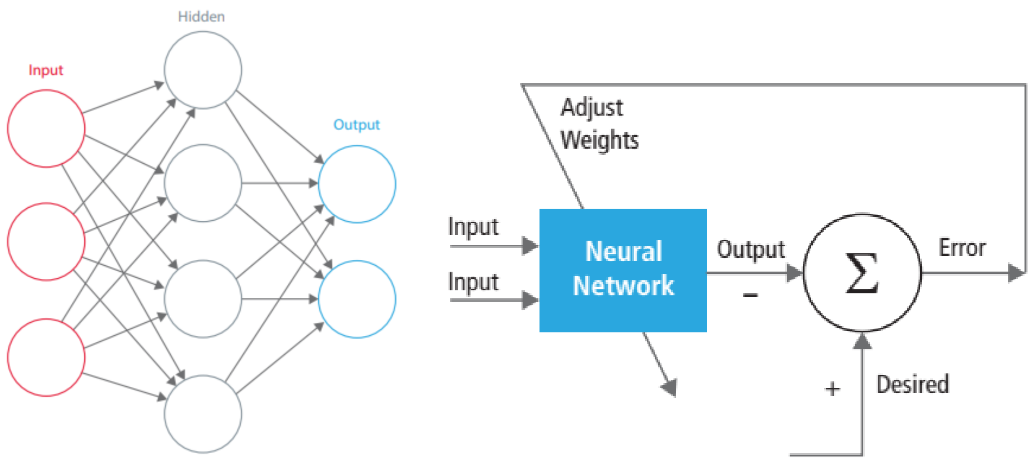


Figure 1: An artificial neural network (left) and training of neural networks (right)

Conv layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulation. Each convolutional neuron processes data only for it's receptive field. Although fully connected put forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a slight (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a significant variable. For an instance, a fully connected layer for a small resolution image of size 100 x 100 has 10,000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, let the network to be deeper with fewer parameters [1]. For instance, regardless of image size, tiling regions of size 10 x 10, each with the same shared weights, requires only 100 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during back propagation in the traditional neural networks are avoided [1].

Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In extension, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the layer priority [1].

With other words convolutional neural networks talks about how the 3-Dimensional convolutional neural network replicate the simple and complex cells of the human brain, including the receptive fields that humans experience through their senses. The ConvNet structure and what their biological connection is, and the optimum functionality which can be extracted from them [2].

Like the way our brains simple identify objects when we see a picture, the goal is to get computers to recognize objects in the same manner. However, there exists a huge difference between what a human brain sees when looking at an image or a computer. To a computer, an image is just another array of digits. Each object has it's own pattern and that is what the computer will use to identify an object in an image. To explain convolutional neural networks in simple terms: just like parents train their children to understand what a ball is or what food is or what is good and evil similarly, computers are also trained by showing a million images of the same object so that their ability to recognize that object increases with each sample trained from the data set [2].

The true catching on of CNNs came with Alex Krizhevsky winning 2012's ImageNet competition wherein he used the networks to drop the image classification error from 26% to 15%. This was a substantial drop and was considered a turning point in the history of digital image classification. Since then, several digital giants have used CNNs in functionalities that will help their business grow such as Google, Amazon, Instagram, Facebook, and Pinterest. CNNs are structured differently as compared to a regular neural network. In a regular neural network, each layer consists of a set of neurons. Each layer is connected to all neurons in the previous layer. The way convolutional neural networks work is that they have 3-Dimensional layers in a width, height, and depth manner. All neurons in a particular layer are not connected to the neurons in the previous layer. Instead, a layer is only connected to a small portion of neurons in the previous layer [2].

The top layer is understood as the mathematical layer. It is essentially the conv layer and deals with understanding the number pattern it sees. Let's assume the first position in this layer starts applying a filter around the top left corner of the image. The filter is also referred to as a neuron or a kernel. It reads that part of the image and forms a conclusion of an array of numbers, multiplies the array, and deduces a single number out of this process. This single number represents the top left corner that the convolutional layer has just read of the image. The part of the image that the filter scans over is the receptive field. The filter then moves right by only one unit and starts the same process again. In this style, the convolutional layer reads the entire image and assigns a single number to each unit. This data gets stored in a 3D array. In essentiality, this entire process functions like the human brain. What we are referring to as the receptive field in the world of CNNs is the visual field in the world of human biology. [2]

The next layer encountered is the Rectified Linear Unit Layer (a.k.a. ReLU). This is where the activation functions take place. The activation function is initially set a zero threshoold. The activation gradient only functions at 0 and 1 and does not include intermediary gradients like it’s predecessors. Due to its linear, non-saturating form, it is said that ReLUs greatly aide in the declining gradient of error. However, due to the fragile nature of a ReLU, it is possible to have even 40% of your network dead in a training dataset.

At a higher level, the first layer in a deep convolutional neural network is the convolutional layer, followed by a rectified linear unit, followed by another convolutional layer, and then alternating rectified linear units and pool layers in conjunction with only one more convolutional layer. While the process that the first convolutional layer follows is pretty straight forward, the process gets more complex and extremely complex as we go down layers, as the convolutional layers are no longer dealing with a simple image. They are dealing with the processed output of the initial mathematics applied at each level [2].

As is with any completed product, its required to have one final layer encompassing all the interior complexities. This layer is the completion layer in a convolutional neural network, his name is “the fully connected layer”. It takes the final output of the layer before it (be it a ReLU or a convolutional layer) and provides an N-Dimensional vector as output. ‘N' here represents the number of classes the program chooses from. For a short example, if the program is looking at pictures of horses, it will look at high-level features such as 4 legs, the hooves, or the tail, or muzzle. This fully connected layer will look at the high-level features and connect that with the image thus giving the output of a classification of a horse [2].

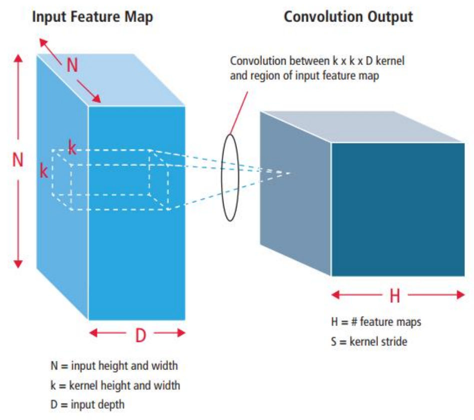

Input Feature Map
Convolution Output
Convolution between k x k x D kernel and region of input feature map
N
k
k
D
H
H = # feature maps
S = kernel stride
N = input height and width
k = kernel height and width
D = input depth

Figure 2: Representation of convolutional process

Importance and practical applications of this algorithm

While neural networks and other pattern based detection methods have been around for the past 60-70 years ago, there has been significant development in the area of convolutional neural networks in the recent past. This section covers the advantages of using CNN for image recognition with broad usage spanning across automotive, industrial, medicine, robotics, agriculture, computer science and so on.

Some of the key applications of CNN are listed below:

1. Decoding Facial Recognition

Facial recognition is broken down by a convolutional neural network into the following major components:

- Identifying every face in the picture;
- Focusing on each face despite external factors, such as light, angle, pose, etc.;
- Identifying unique features;
- Comparing all the collected data with already existing data in the database to match a face with a name. A very similar process is followed for scene labeling as well [2].

2. Analyzing Documents

Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major post in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute. It is said with the use of CNNs and newer models and algorithms, the error rate has been go down to a minimum of 0.4% at a character level, though it's complete testing is yet to be widely seen [2].

3. **Historic and Environmental Collections**

CNNs are also used for more complex purposes such as natural history collections. These collections act as key players in documenting major parts of history such as bio diversity, evolution, habitat loss, biological invasion, and climate change [2].

4. **Understanding Climate**

CNNs can be used also to play a major role in the fight against climate changes, especially in understanding the reasons why we see such drastic changes and how we could experiment in curbing the effect. It is said that the data in such natural history collections can also provide greater social and scientific insights, but this would require skilled human resources such as researchers who can physically visit these types of repositories. There is a need for more power to carry out deeper experiments in this field of domain [2].

5. **Grey Areas**

Introduction of the grey area into CNNs is posed to provide a much more realistic picture of the real world, the people of the cities being surrounded by tall buildings every day most gray or with glass walls. Currently, CNNs largely function exactly like a machine, seeing a true and false value for every question. However, as humans, we understand that the real world plays out in a thousand shades of grey. Allowing the machine to understand and process fuzzier logic will help it to understand the grey area us humans live in and strive to work against. This will help CNNs get a more holistic view of what human sees everyday [2].

6. **Advertising**

CNNs have already brought in a world of difference to advertising with the introduction of programmatic buying and data driven personalized advertising widespread most often in social media or in biggest cities on big screens of the buildings.

7. Other Interesting Fields

CNNs are hold to be the future with their introduction into driverless cars, robots that can mimic human behavior, aides to human genome mapping projects, predicting earthquakes and others natural disasters, and maybe even self-diagnoses of medical problems. So, you would not even have to drive down to a clinic or schedule an appointment with a doctor to ensure your sneezing attack or high fever and not symptoms of some rare disease. One problem that researchers are working on with CNNs is brain cancer detection. The earlier detection of brain cancer can prove to be a big step in saving more lives affected by this illness.

Demo application using CNN algorithm to differentiation between cats and dogs

Let's start with a short description of the application: we have a training set consisting of 25000 sample low resolution images, each image has a corresponding id based on filename, plus a csv file where each id have a label (1 = dog, 0 = cat), of which 80% represent the training phase and the remaining 20% the testing phase.

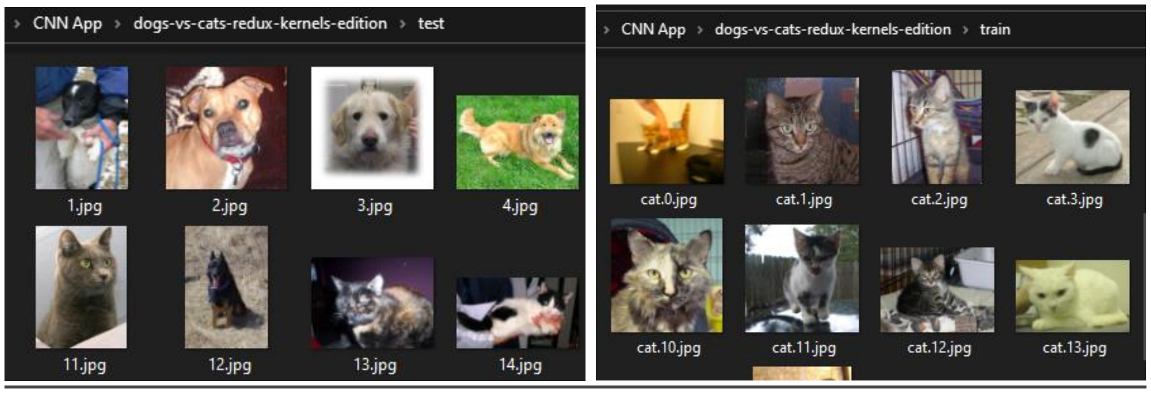


Figure 3: Training set preview

Our expectations are that after completing the training and analysis, software will choose twelve random sample images from the test directory and apply the appropriate label to each image.

First order of business is to convert the images and labels to array information that we can pass through our network. To do this, use a helper function to convert the image name to an array. The convolution layer computes the output of neurons that are connected to local regions or receptive fields in the input, each computing a dot product between their weights and a small receptive field to which they are connected to in the input volume. Each computation leads to extraction of a feature map from the input image. In other words, imagine you have an image represented as a 5x5 matrix of values, and you take a 3x3 matrix and slide that 3x3 window or kernel around the image. At each position of that matrix, you multiply the values of your 3x3 window by the values in the image that are currently being covered by the window. As a result, you will get a single number that represents all the values in that window of the images. It works because of filters, which are multiplied by the values outputted by the convolution.

The objective of subsampling is to get an input representation by reducing its dimensions, which helps in reducing overfitting. One of the techniques of subsampling is the max pooling. With this technique, you select the highest pixel value from a region depending on its size. In other words, max pooling takes the largest value from the window of the image currently covered by the kernel. For example, you can have a max-pooling layer of size 2 x 2 will select the maximum pixel intensity value from 2 x 2 region.

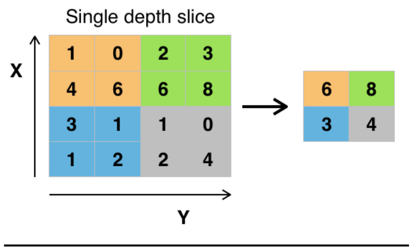
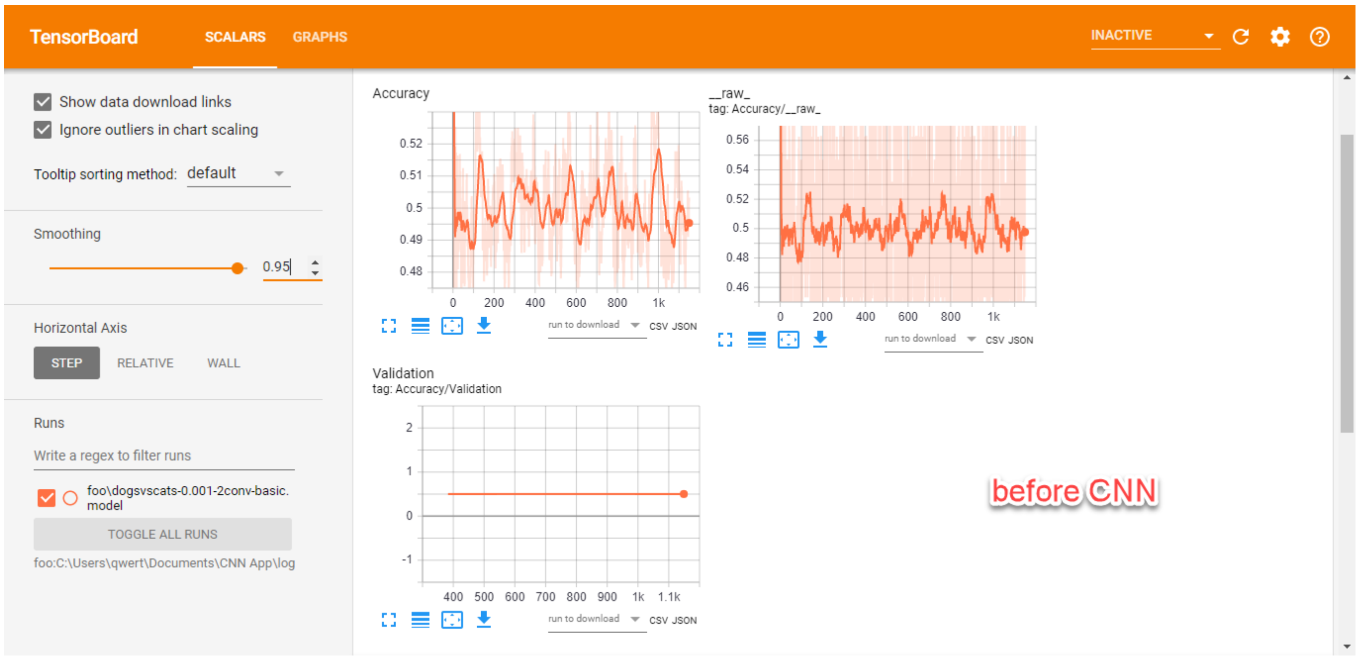
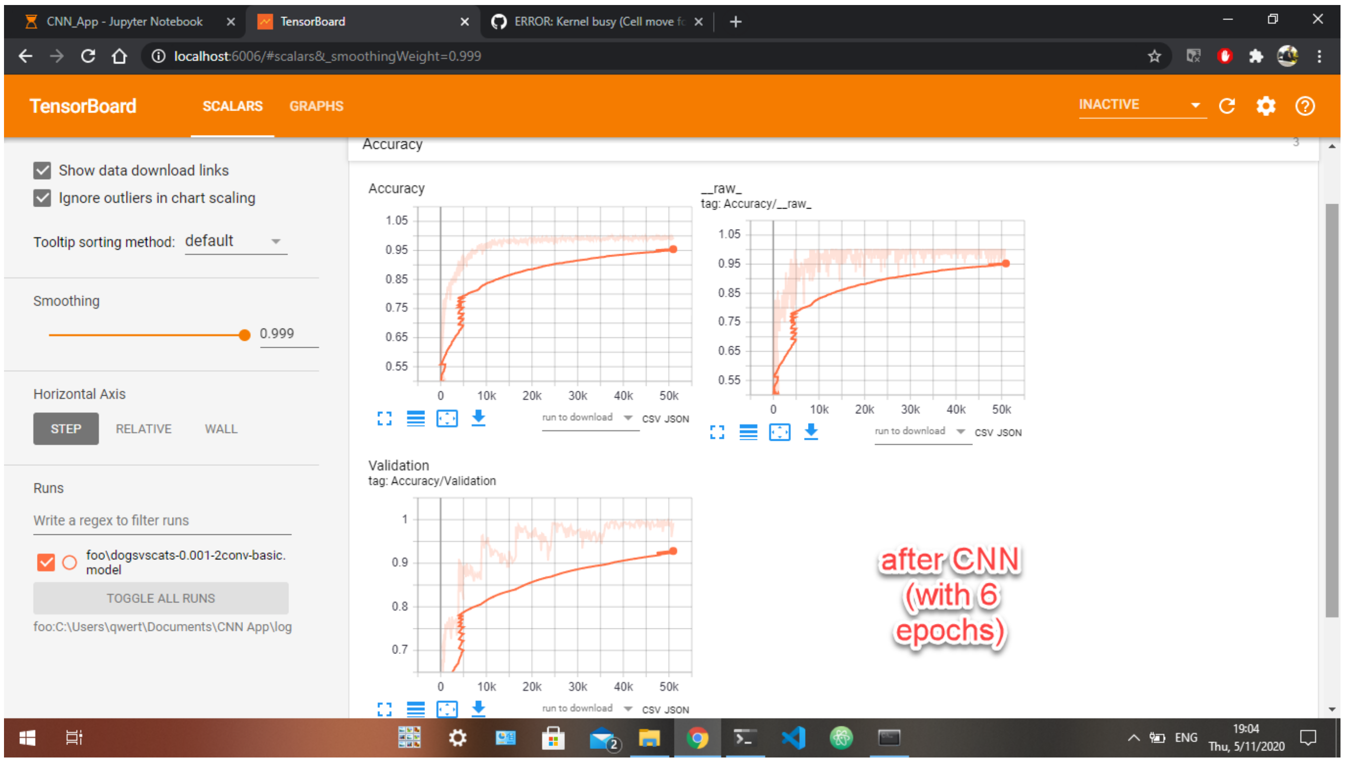


Figure 4: Max pooling

Let’s compare the results before and after implementing convolutional neural networks, first let’s compare graphs generated by tensorboard:



☑ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.95
Horizontal Axis [✓ STEP] RELATIVE WALL
Runs
Write a regex to filter runs
☑ foo\dogsvscats-0.001-2conv-basic.model
TOGGLE ALL RUNS
foo:C:\Users\qwert\Documents\CNN App\log


CNN_App - Jupyter Notebook | TensorBoard | ERROR: Kernel busy (Cell move f

localhost:6006/#scalars&_smoothingWeight=0.999

TensorBoard SCALARS GRAPHS INACTIVE

☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.999

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

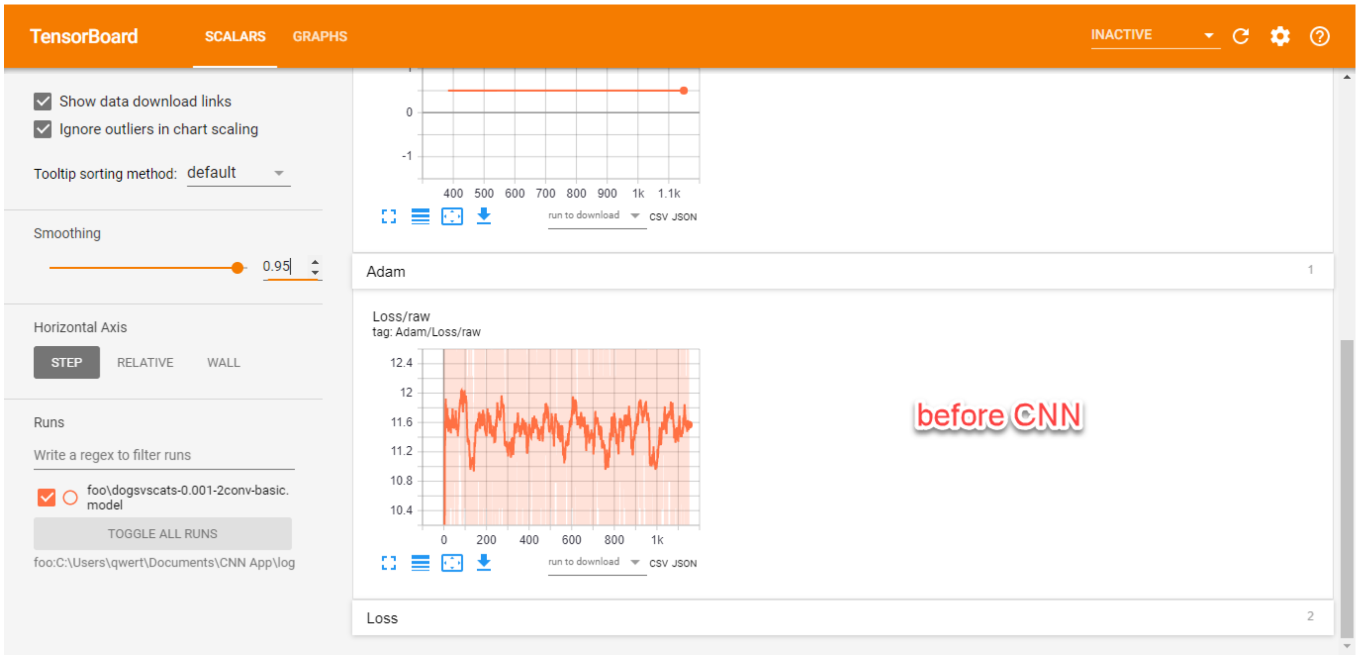
☑ foo\dogsvscats-0.001-2conv-basic.model

TOGGLE ALL RUNS

foo:C:\Users\qwert\Documents\CNN App\log

Accuracy

after CNN (with 6 epochs)


TensorBoard SCALARS GRAPHS INACTIVE

☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.95

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ foo\dogsvscats-0.001-2conv-basic.model

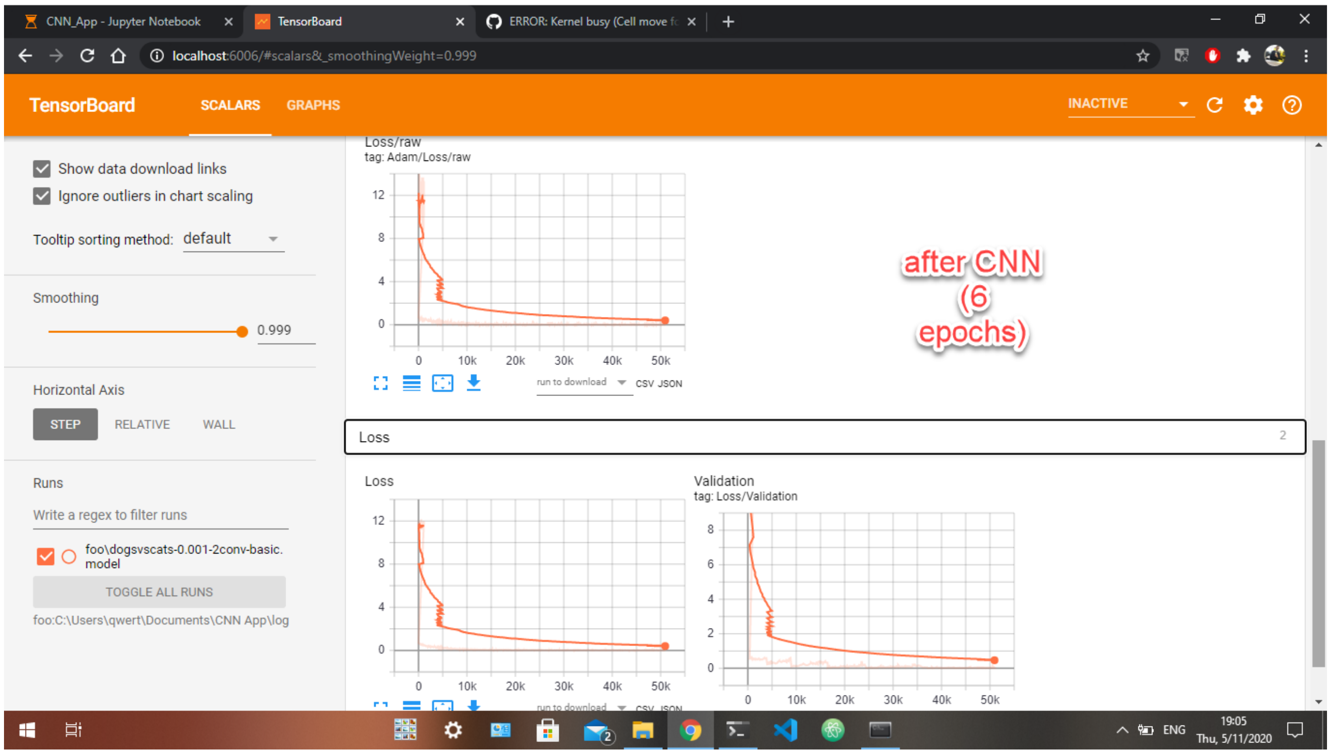
TOGGLE ALL RUNS

foo:C:\Users\qwert\Documents\CNN App\log

Adam

Loss

before CNN



CNN_App - Jupyter Notebook | TensorBoard | ERROR: Kernel busy (Cell move f

localhost:6006/#scalars&_smoothingWeight=0.999

TensorBoard SCALARS GRAPHS INACTIVE

Show data download links ✓
Ignore outliers in chart scaling ✓
Tooltip sorting method: default
Smoothing 0.999
Horizontal Axis: STEP RELATIVE WALL
Runs
Write a regex to filter runs
foo\dogsvscats-0.001-2conv-basic.model
TOGGLE ALL RUNS
foo:C:\Users\qwert\Documents\CNN App\log

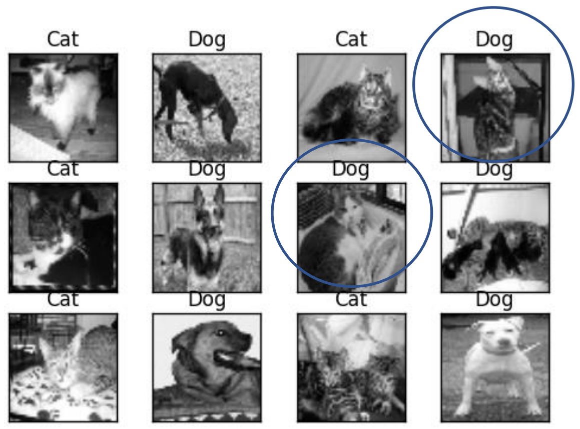
Loss/raw
tag: Adam/Loss/raw

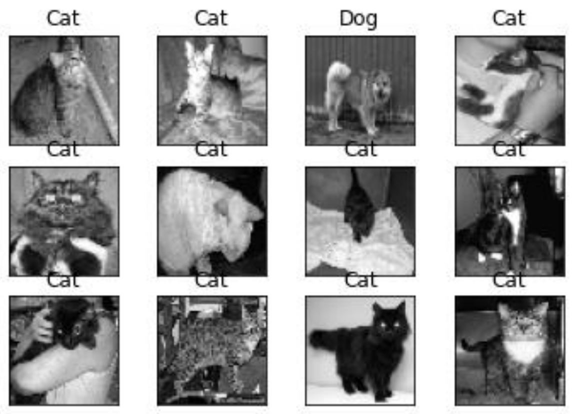
Loss 2

Loss

Validation
tag: Loss/Validation

It’s obvious that graphs with convolutional neural networks implemented are look better, but the second part it’s the final results :





In left side we have twelve completely random samples choosed from test directory without any convolution neural networks applied, we can observe some mistakes in two images, the simple neural network algorithm says “it’s a dog” but actually it’s a cat, we can identify “dog posture” where that cat sits, this is also true for the second mistake. The images are coverted to black and white from color for a better analysis of contours and segmentation.

The CNN algorithm implemented with six epochs have great results as you can see in the right side, all sample images choose randomly were labeled correctly.Here it's a short explication of "epochs meaning": in terms of artificial neural networks, an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs. In other words, if we feed a neural network the training data for more than one epoch in different patterns, we hope for a better generalization when given a new "unseen" input (test data). An epoch is often mixed up with an iteration. Iterations is the number of batches or steps through partitioned packets of the training data, needed to complete one epoch. Heuristically, one motivation is that (especially for large but finite training sets) it gives the network a chance to see the previous data to readjust the model parameters so that the model is not biased towards the last few data points during training. there is no guarantee a network will converge or "get better" by letting it learn the data for multiple epochs. It is an art in machine learning to decide the number of epochs sufficient for a network.

Be aware that there is no guarantee a network will converge or "get better" by letting it learn the data for multiple epochs, a lot of epochs will need a lot of computational power and time to be processed. It is an art in machine learning to decide the number of epochs sufficient for a network. In parallel, when we apply this to other areas of machine learning such as reinforcement learning, we see that an agent may not take the same route to complete the same task. This is because the agent is learning which decisions to make and trying to understand the consequences of such action(s). With a neural network, the goal of the model is generally to classify or generate material which is right or wrong. Thus, an epoch for an experimental agent performing many actions for a single task may vary from an epoch for an agent trying to perform a single action for many tasks of the same nature. In reinforcement learning terminology, this is more typically referred to as an episode.[3]

Given the complexity and variability of data in real world problems, it may take hundreds to thousands of epochs to get some sensible accuracy on test data. Also, the term epoch varies in definition according to the problem at hand. As a specific example of an epoch in reinforcement learning, let's consider traveling from point A to point B in a city. Now, we can take multiple routes to reach B and the task is to drive from A to B a hundred times. Consider an epoch to be any route taken from a set of available routes. An iteration on the other hand describes the specifics of the route like which turns or how many stops. In the reinforcement learning terminology, an iteration is often called an action.[3]

References

https://deepai.org/machine-learning-glossary-and-terms/epoch [3]

https://en.wikipedia.org/wiki/Convolutional_neural_network [1]

https://www.flatworldsolutions.com/data-science/articles/7-applications-of-convolutional-neural
networks.php [2]