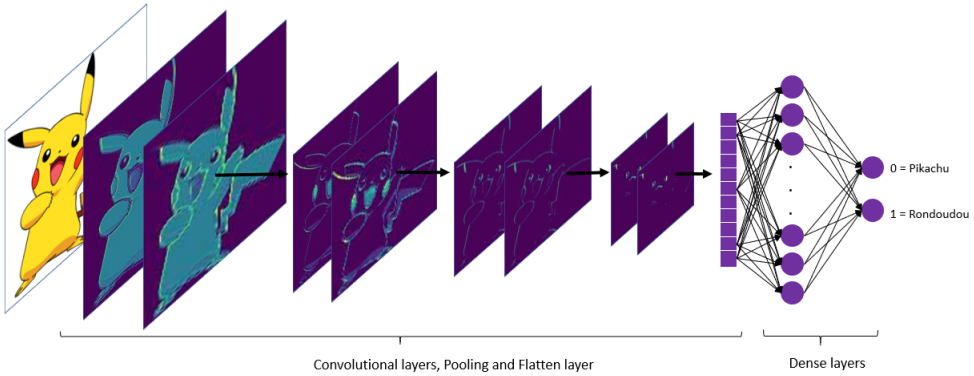
Experimental Design for Machine Learning on Multimedia Data

Oscar Poels, Alexandre Lalau, Alexis Tonneau

Sunday, December 19^{th} 2021

Introduction

The machine learning project we analyze is taken from an existing project available at the following link: `https://github.com/anisayari/Youtube-apprendre-le-deeplearning-avec-tensorflow/tree/master/\%234\%20-\%20CNN`. We have completely redesigned it in order to better analyze the algorithm and to manipulate the information. The machine learning project we are using is a CNN with the purpose of recognizing two types of images: images of the pokemons pikachu and jigglypuff. Our work is available here: `https://github.com/OscarPoels/ML-Berkeley/tree/main`. Our machine learning model consists of image recognition between a pikachu and a jigglypuff. For this, we use the Tenserflow library and the code is written in python. Our network is composed of two main structures, a CNN and a dense layers network.



1 What is the variable the machine learner is supposed to predict? How accurate is the labeling? What is the annotator agreement (measured)?

The variable the machine learner is supposed to predict is pikachu or jigglypuff. After labeling the images ourselves by putting pikachu's images in the same folder and jigglypuff's images in another folder, the labeling accuracy is 100%. We only have one annotator, which is the hand-labeling process, so we don't need to measure the agreement of annotators.

2 What is the required accuracy metric for success? How much data do we have to train the prediction of the variable? Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do we expect bias?

The precision metric required to have a satisfactory result must be $G > 1$. This would mean that our model generalizes. We have a total of 711 images that belong to 2 different classes, 469 are pikachu images and 242 for jigglypuff. 569 files are used for training and 142 for validation. The classes are not balanced, in fact, the pikachus represent 66.05% of our dataset while the jigglypuffs represent 33.95%. This is due to the fact that the data concerning the pikachu are much easier to find. We have three modalitites that we are exploiting which is the RGB of the images. The temporal information we have is the execution time that the algorithm takes on a google cloud virtual machine. The program takes about 39 seconds to run including the Convolution Layers.We expect some bias because our classes are unbalanced, and the amount of data might not be enough. Because our dataset is not that large, we expect a fairly large noise.

3 What is the Memory Equivalent Capacity for the data (as a dictionary). What is the expected Memory Equivalent Capacity for a neural network?

We compute the equivalent capacity of the memory using the number of thresholds. We calculate this number according to our model by using a dummy network and computing the number of thresholds that the dummy network requires. This is 131072 thresholds, which represents a MEC of 17 bits for the data.

We can verify our calculation with the brainome library which confirms that we have this MEC.

$$\log_2(131072 + 1) = 17\, bits \tag{1}$$

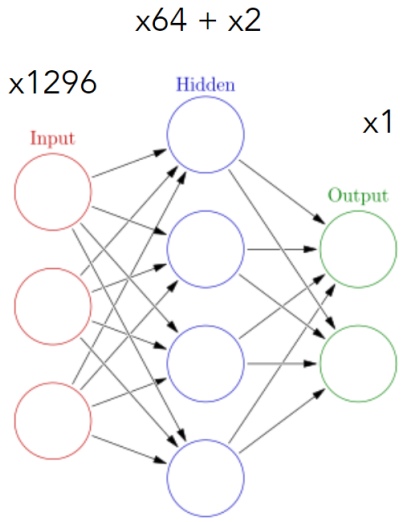
.

$$Model\ Equivalent\ Capacity\ for\ data\ (MEC):\ 17\, bits \tag{2}$$

.

Our model is composed of several convolution layers with different filters. After going through all these layers, the neural network takes 1296 inputs, it then go through a neural network which is composed of 1 hidden Layer of 64 neurons and it final ends in 2 outputs. This makes a total of 83008 parameters. Each parameter is a Byte = 8 bits. Indeed,

$$MEC\,for\,neural\,network = (1296 \times 8 + 1) \times 64 = 663616 + 64 = 663\,680\,bits \tag{3}$$



4 What is the expected generalization in bits/bit and as a consequence the average resilience in dB? Is that resilience enough for the task? How bad can adversarial examples be? Do we expect data drift?

In order to compute our generalization we need the number of correctly classified instances. Here is our method to get this number:

$$\begin{aligned} \#correctly\,classified\,instances &= Accuracy \times \#Images\,involved \\ &\quad \times \#Inputs \times Conversion\,in\,bits \end{aligned} \tag{4}$$

Which makes:

$$\begin{aligned} \#correctly\,classified\,instances &= 0.957 \times 711 \times 1296 \times 8 \\ &= 7\,054\,667\,bits \end{aligned} \tag{5}$$

Now, we can get the generalization:

$$\begin{aligned} Generalization &= \frac{\#correctly\,classified\,instances}{MEC} \\ &= \frac{7054667}{663680} \\ &\approx 10.63\,bits/bit \end{aligned} \tag{6}$$

According to the metrics that we have computed, we can now calculate the resilience:

$$\begin{aligned} Resilience &= 20\log_{10}(Generalization) \\ &= 20\log_{10}(10.63) \\ &\approx -20.54dB \end{aligned} \tag{7}$$

We decided to express the number of correctly classified instances in bits to make the calculation of the generalization homogeneous. Indeed we divide this number by the MEC which is expressed in bits.

Our resilience, being -20.54 dB is enough because we find out that in average, tensorflow estimates the noise of our model around 5.54 dB (computing with our code) so, $\mid Resilience \mid > \mid Noise \mid$.

Our model is to analyze an image to predict whether it is a pikachu or a jigglypuff, so our adversarial example is a wrong prediction. It is when we predict pikachu for jigglypuff and vice versa. So it is considered as a huge problem but it happens very rarely thanks to the precision of our model which is very high. And, we don't expect data drift.

5 Is there enough data? How does the capacity progression look like?

This question is related to question in section **3)** on accuracy, which we had left open. Let's start by answering it, **What is the required accuracy metric for success?**

To have a sufficient precision, as we already said we need $G > 1$. With the previous results,we would need:

$$\begin{aligned} Generalization &= \frac{\#correctly\,classified\,instances}{MEC} \\ &> 1 \end{aligned} \tag{8}$$

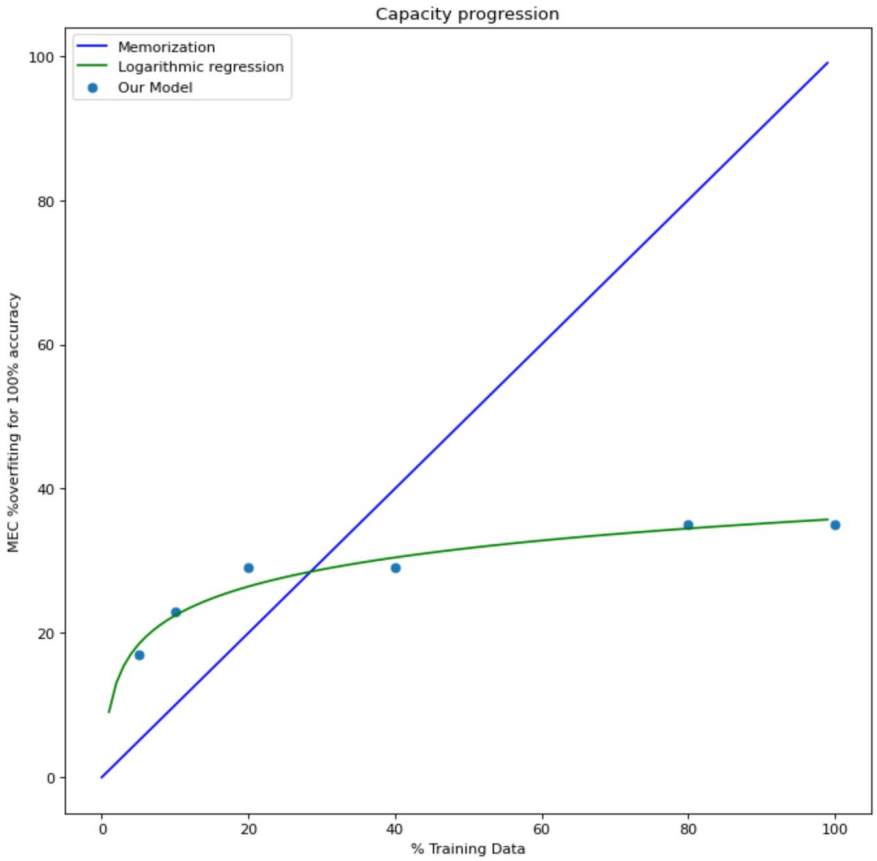
Which means:

$$\begin{aligned} \frac{Accuracy \times 771 \times 1296 \times 8}{663680} &> 1 \\ Accuracy &> \frac{663680}{771 \times 1296 \times 8} \\ Accuracy &> 0,09 = 9\% \end{aligned} \tag{9}$$

This shows that the amount of data we are injecting is so small compared to our MEC that our accuracy does not need to perform well. On the other hand, if we want to optimize our model, we have to assume that the MEC can be lower. This is why we will be satisfied with an accuracy of 85% in order to be consistent with the accuracy of other CNNs.

Since our generalization is positive, the data we have is sufficient. And our model is not overfiting

After calculating the capacity progression at 5%, 10%, 20%, 40%, 80%, 100% for our system and after doing a logarithmic regression, we get the following results:



This confirms us in the idea that a rule has been found for our model.

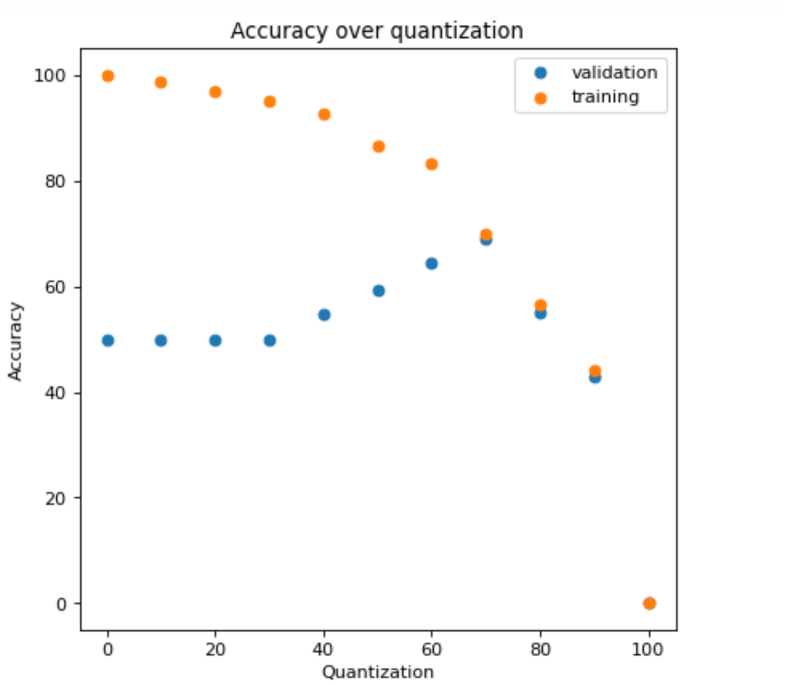
6 Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% memorization? If not, why (diagnose)?

According to the tests we have performed, we end up with 100%. It of course, implies that our system is overfiting.

```
Epoch 23/30
12/12 [==============================] - 7s 427ms/step - loss: 0.0119 - accuracy: 0.9947 - val_loss: 0.0371 - val_accuracy: 0.9859
Epoch 24/30
12/12 [==============================] - 7s 429ms/step - loss: 0.0120 - accuracy: 0.9965 - val_loss: 0.1639 - val_accuracy: 0.9718
Epoch 25/30
12/12 [==============================] - 7s 429ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.0745 - val_accuracy: 0.9718
Epoch 26/30
12/12 [==============================] - 7s 433ms/step - loss: 4.5204e-04 - accuracy: 1.0000 - val_loss: 0.2546 - val_accuracy: 0.9648
Epoch 27/30
12/12 [==============================] - 7s 432ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.1273 - val_accuracy: 0.9718
Epoch 28/30
12/12 [==============================] - 7s 431ms/step - loss: 4.2954e-04 - accuracy: 1.0000 - val_loss: 0.1431 - val_accuracy: 0.9718
Epoch 29/30
12/12 [==============================] - 7s 434ms/step - loss: 1.9869e-04 - accuracy: 1.0000 - val_loss: 0.1749 - val_accuracy: 0.9718
Epoch 30/30
12/12 [==============================] - 7s 432ms/step - loss: 1.0705e-04 - accuracy: 1.0000 - val_loss: 0.1964 - val_accuracy: 0.9718
<keras.callbacks.History at 0x7fda70407e50>
```

7 Train your machine learner for generalization: Plot the accuracy/ capacity curve. What is the expected accuracy and generalization ratio at the point you decided to stop? Do you need to try a different machine learner? How well did your generalization prediction hold on the independent test data? Explain results. How confident are you in the results?

Ater ploting our results, this is what we get:



According to our model and this curve the accuracy that we expect is 66% and with a generalization ratio of 22,2 bits/bit.

Here is the calculation of the generalization ratio:

$$\begin{aligned} & Generalization\ (N\ approx) \\ & = \frac{\#correctly\ classified\ instances}{MEC} \\ & = \frac{0.66 * 1296 * 711 * 8}{219014} \\ & = \frac{4865287}{219014} \\ & \approx 22,2\ bits/bit \end{aligned} \tag{10}$$

We decided to stop our model at this point during our process of quantization because it is when the validation set and the training set intersect.

Having a model that we are satisfied with, relying on its accuracy for example, we think we do not need another machine learner.

The generalization prediction was verified because the tests we performed with a lower generalization gave us an accuracy of 66% and a lower MEC. This can be explained with the quantization process (reducing the model size).

We are aware that our model will allow for a more efficient generalization. On the other hand, the accuracy being low (66%), we think that this kind of model is not reliable and is not what we are looking for. This comes back to a previous question which tended to ask us about the value of the accuracy that we would like. In conclusion, our model could be more "profitable" but less efficient.

On the independent test data, our results were found to be consistent with the predictions we made, we achieved similar results with the accuracy and MEC we expressed earlier. We can therefore be confident in the results we obtained and affirm that our model is reliable

8 Comment on any other quality assurance measures possible to take/ the authors should have taken. Are there application-specific ones? If time is present: How did you deal with it?

We could also have taken into account time as a measure, the latter being also linked to the computing power of the machines we used. Indeed, since

we chose to work personally on this project and to develop it, we were limited by the data at first and then by the computing power which forbade us to have too important inputs in our neural network. We could have quantified this by analyzing the time curves as a function of the computing power and the computing power as a function of the MEC and its efficiency. For what we were able to achieve with what we have, we chose models that meet these limits, allowing us to wait only a few minutes at most for some runtimes.

9 How does your experimental design ensure repeatability and reproducibility?

We have tried as much as possible to make our project repeatable and reproducible, however some aspects of our system may be problematic. First of all, we would like to define the terms used to avoid any confusion:

Artifact: By "artifact" we mean a digital object that was either created by the authors to be used as part of the study or generated by the experiment itself. For example, artifacts can be software systems, scripts used to run experiments, input datasets, raw data collected in the experiment, or scripts used to analyze results.[1]

Repeatability (Same team, same experimental setup): The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.[1]

Reproducibility (Different team, same experimental setup): The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts.[1]

Replicability (Different team, different experimental setup): The measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.[1]

We will detail the different parts of the project according to the ACM badges as follows:

Artifacts Evaluated - Functional Documented: We have given the library we used for our machine learning model, we have also explained how it works in brief, as well as given the source of the code and the part of the program we modified, everything is available on Github. On the other hand, not being the objective of our report, we did not clearly explain step by step the process we used to allow the use of brainome for example. But it is still reproducible from the links we have provided.

Consistent: The artifacts we have used are quite consistent with the results of our research, we have been able to detail the calculations performed and put forward the results obtained using our system.

Complete: Once again, all relevant components are included in our document and allow the reproducibility of the results.

Exercisable: This is the only problem that can be encountered unfortunately, in fact, the libraries that we use are always maintained, which leads them to change and may not allow to reproduce all the results obtained. We also used a virtual machine under Google Colab to carry out our project and we know that a certain computing power is necessary to the realization of our system. Some machines may also not be compatible with the code we provided. Fixing these problems would take a lot of time, and since we think that the theoretical side of our work was more important, we preferred to focus on that.

As far as availability is concerned, everything is available on Github as we explained, publicly and at any time. So we think that the availability of the artifacts is respected. Our results should be replicable, indeed, we could test and ask people outside of our project to install and initialize our model simply with the links provided. As far as the results are concerned, if the system set up is the same as the one we made for this study, the results should be the same, which would guarantee the replicability

Statements of Contribution

Alexis took care of initializing the project by answering the first question. Alexandre and Oscar worked equally on the rest of the questions. Moreover, Oscar modified the source code that we had chosen in order to manipulate it, plot the data and make it readable for Brainome.

References

[1] *Artifact Review and Badging Version 1.1* https://www.acm.org/publications/policies/artifact-review-and-badging-current

August 24, 2020.