Internship Report BUT 2

GODINAT Caetano

August 27, 2024

Exploring Machine Learning with Hasktorch

Ochanimizu University
BEKKI LAB
FACULTY OF SCIENCES
2-1-1 ohtsuka, bunkyo-ku
112-8610 TOKYO
JAPON

8th April - 14 June

Reference professor : Mr Bruno Mery

Host organization tutor : Bekki-sensei

1 Acknowledgments

Before the abstract I first want to thank Mr. Mery and Mrs. Cartier, for providing me with this unique opportunity to represent the IUT of Bordeaux-Gradignan. I am especially thankful to Bekki-sensei, who graciously welcomed me into his lab for the duration of my internship. I also wish to extend my sincere appreciation to Sora-sensei, who led the seminars and supervised my internship. Her supervision and insightful feedback greatly enhanced my learning experience. My heartfelt thanks go to the other students in the laboratory, Tanaka-san and Lachlan. Your camaraderie and support were essential in making my internship enjoyable and productive. Thank you all for creating a warm and collaborative environment. It was a pleasure to be part of such a dedicated and friendly team. And of course a thanks to Josselin who was working on the same subject as me.

2 Abstract

The BekkiLab is specialized in mathematical linguistics and as machine learning is becoming more and more used, tools like Hasktorch a Haskell library for machine learning are becoming more refined. The objective of this internship was learning and experimenting with this library, by developing models while make improvements to their complementary library Hasktorch-tools. The approach was to try new things and testing the limits while learning how the libraries are structured. I ended up making improvements in the hasktorch-tools library and learned how to design, train and evaluate machine learning models.

3 Résumé

Le BekkiLab se spécialise en linguistique mathématique et, avec l'utilisation croissante de l'apprentissage automatique, des outils comme Hasktorch, une bibliothèque Haskell pour l'apprentissage automatique, deviennent de plus en plus raffinés. L'objectif de ce stage était d'apprendre et d'expérimenter avec cette bibliothèque, en développant des modèles tout en améliorant leur bibliothèque complémentaire Hasktorch-tools. L'approche consistait à essayer de nouvelles choses et à tester les limites tout en apprenant comment les bibliothèques sont structurées. J'ai fini par apporter des améliorations à la bibliothèque Hasktorch-tools et j'ai appris à concevoir, entraîner et évaluer des modèles d'apprentissage automatique.

Contents

4 Internship Context

4.1 Ochanomizu University

[1]

Figure 2: Ochanomizu University

Established in 1875, Ochanomizu University is Japan's first institution of higher education for women. Originally located in Ochanomizu, it moved to Otsuka Bunkyo-ku after the Great Kanto Earthquake in 1923. The university's goal is to empower women globally to combat gender inequality. In 2004, it became a national university corporation, opening its doors to more women, including those from overseas and developing countries. It actively supports the Sustainable Development Goals and is affiliated with schools from kindergarten to high school, providing comprehensive assistance to students throughout their academic journey. In May 2021, Ochanomizu University had 2,807 students enrolled in a range of study areas, including scientific, artistic, and literary disciplines.

4.2 Bekki Lab

Figure 3: BekkiLab

Josselin and I are tied to the BekkiLab as student researchers under the guidance of Mr. Bekki, who is the director of the lab. The Bekki Lab, affiliated with Ochanomizu University, focuses on the study of mathematical linguistics, specializing in Combinatory Categorial Grammar (CCG) and Dependent Type Semantics (DTS). The structure of the lab is straightforward: Mr. Bekki is the Director, Ms. Tanaka is the secretary, and Ms. Sora, a former student now working at Google Japan, supervises our internship activities due to her extensive experience with Hasktorch. Additionally, there are nine other students ranging from second-year undergraduates to second-year Master's students, including Ms. Mizuki, Ms. Hanaka, Ms. Asa, Ms. Aoi, Ms. Mai, Ms. Kana, Ms. Koharu, Ms. Nanako, and Ms. Kotomi. The Bekki Lab is funded by Ochanomizu University and the Japanese government for research purposes. Mr. Bekki not only supports his students but also promotes their projects and conducts research in computational linguistics, focusing on the integration of machine learning and linguistics. Ms. Tanaka serves as the lab's secretary, though

I didn't have the opportunity to ask her about her specific duties. All the students in the lab are engaged in learning and researching topics related to CCG and DTS. The work environment is friendly and open-minded, encouraging questions and interactions. Our schedule has been flexible from the start. Mr. Bekki requires our presence in the lab only for a weekly meeting, typically held on Tuesdays, where we receive exercises to deepen our understanding of specific concepts and provide feedback on our weekly progress. However, we usually come to the lab when others are present, typically Monday through Thursday, and work remotely on Fridays.

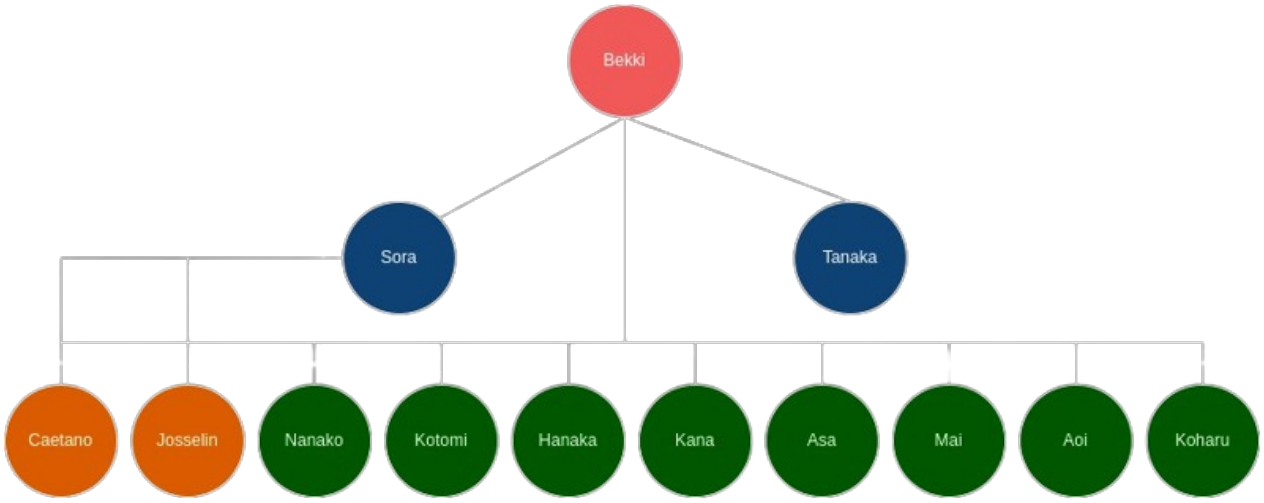


Figure 4: Hierarchy of the laboratory

4.3 Hasktorch

For the subject of my research, Mr. Bekki asked us to explore the Haskell library Hasktorch. Hasktorch is based on Libtorch, which is coded in C++ and is also used to develop PyTorch, one of today's most popular Python modules for developing neural networks. Although Hasktorch is still in the early stages of development, it shows great promise in improving the efficiency of coding neural networks. Today, we can view neural networks almost as a new programming paradigm, with a set of mathematical functions (such as model, inference, training, etc.) that can be used to generate data. Hasktorch significantly contributes to coding paradigms by enabling more function composition in neural networks. This means that we can create small blocks that can be assembled in various configurations. It also supports Abstract Data Types, allowing us to create more generic functions. Together, these features make it much easier to create abstract and adaptable models. [11]

5 Internship

5.1 Hasktorch Installation

Installing Hasktorch can be challenging for Windows and MacOS users as it requires extensive file manipulation and configuration. I use a Linux distribution, so this issue was easier to manage, but without proper documentation, it is still easy to get lost. Since Hasktorch is still a relatively small library used by few people, there are numerous minor issues that can be specific to each work environment. We also used a library developed by the BekkiLab called Hasktorch-tools, which offers useful tools for working with Hasktorch, and its installation process is similar to Hasktorch's. The project configuration is managed with the stack.YAML and packages.YAML files.

5.2 Principles of Hasktorch and machine learning

It will be challenging to explain what I did and learned during this internship without delving into the basic principles of neural networks. Therefore, here is a brief introduction to the topic: Machine learning is a method of data analysis that automates the process of building analytical models. There are various machine learning techniques, each suited to creating different models for solving different types of problems. In simple terms, a model can be defined as a function that takes input data and produces a prediction, classification, or other output. At the core of every neural network are neurons, which have the following structure:

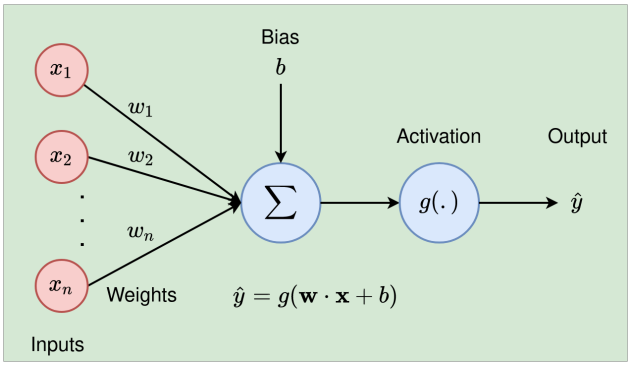


Figure 5: Neuron
[5]

Essentially, a neuron receives inputs, multiplies each input by its respective weight, sums all the results, adds the bias, passes the sum through an activation function, and returns the result. Using these neurons, we can construct a neural network, typically structured like this:

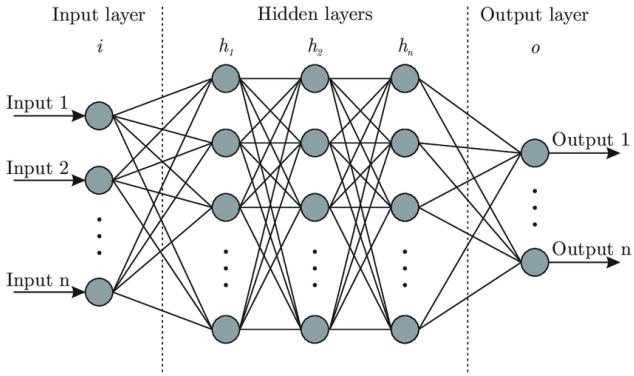


Figure 6: Neural Network
[9]

The activation function serves to compress the results, transforming the range from negative infinity to positive infinity into a more manageable range, typically between 0 and 1. Here's an example of a simple activation function, the sigmoid function:

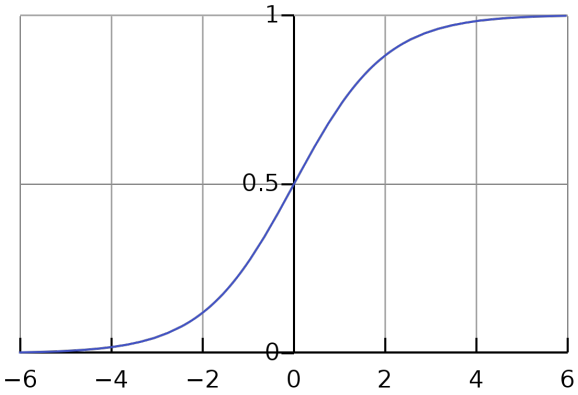


Figure 7: Sigmoid Function
[3]

Machine learning revolves around optimizing weights, a process achieved through the manipulation of matrices. Hasktorch, accordingly, operates on these N-dimensional matrices known as tensors.

5.3 First steps

5.3.1 Linear Regression

Initially, I was instructed to begin by developing a model to solve a linear regression problem. These problems can be addressed using linear functions.

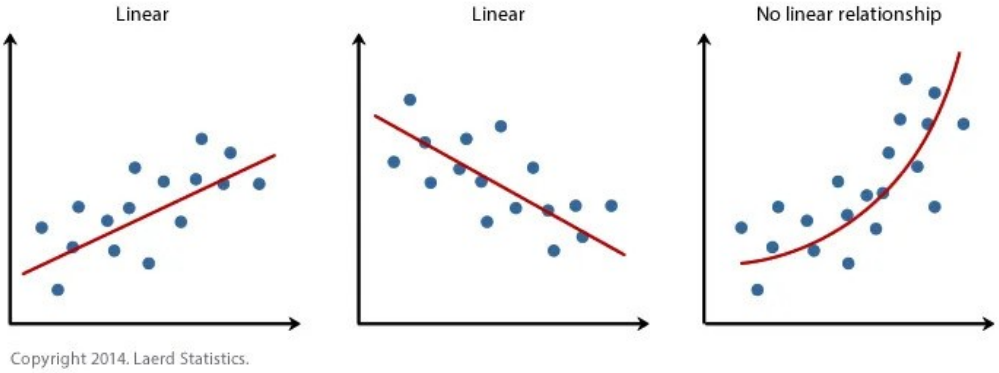


Figure 8: Linear Regression
[7]

Solving a simple problem using machine learning served as a practical exercise to learn about setting up a training loop and understanding hyperparameters. Hyperparameters, unlike the

trainable parameters of the neural network, remain constant during training. These parameters can be adjusted to enhance the model's learning efficiency, accuracy, and generalization to new data, among other properties. Here are some examples of hyperparameters:

- Learning rate - learning speed of the model
- Number of Epochs - number of training loops
- Loss Function - function that will calculate according to the expected output and the output returned by the model the difference between both.
- Optimizer - function that will optimize the learning rate according to the loss
- Batch size - number of samples used in one forward or backward
- number of hidden layers
- number of neurons inside each layer
- the activation function for each layer

5.3.2 Training

Before training the model, the data needs to be processed to make it understandable to the computer. This involves converting non-numeric values into numeric ones, normalizing the data to ensure each value has a consistent impact on calculations, and removing columns with insufficient data or little relevance to the training process. In the example below, you can see the transformation of one of the datasets from its original state to its preprocessed form.

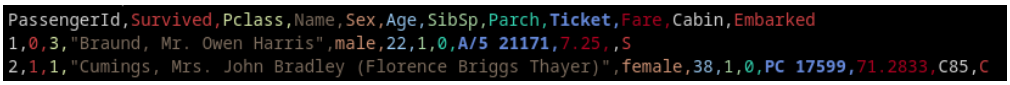


Figure 9: Before

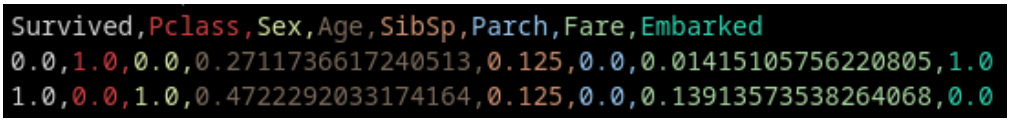


Figure 10: After

In the example provided, we can observe that non-numeric data, such as "sex," has been transformed into numerical values (0 for female, 1 for male). Additionally, the data has been normalized to a range between 0 and 1. Columns deemed irrelevant or containing a significant number of missing values, such as "Cabin," have been removed. For columns with a few missing values, such as "Age," the gaps have been filled with the mean value of the column. Furthermore, the data needs to be separated into two sets: the training set and the validation set. Typically, around 90% of the data is allocated to the training set, with the remaining 10% reserved for validation. However, this ratio can vary based on the dataset's size; more data generally results in a smaller validation set to ensure adequate training. This separation is crucial for assessing whether the model is overfitting the data. Testing on unseen validation data helps evaluate the model's generalization capabilities.

5.3.3 Learning

The learning rate is a hyperparameter that determines the size of the learning steps during the training process. Choosing the correct learning rate is crucial, as it significantly impacts the learning process, as illustrated in the following image:

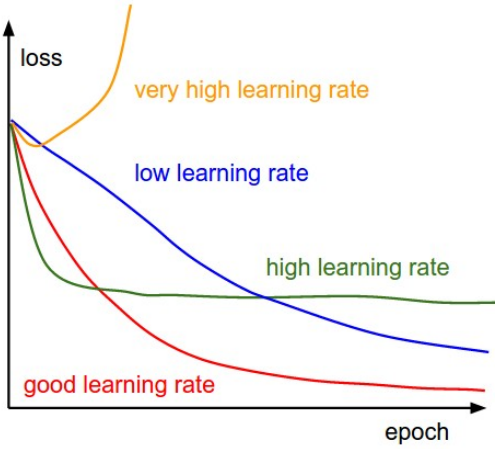
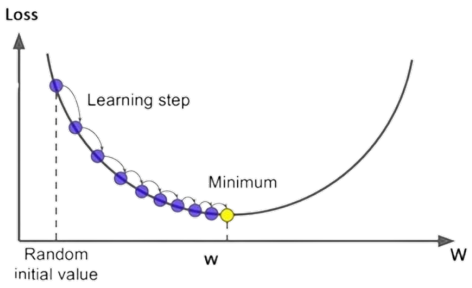
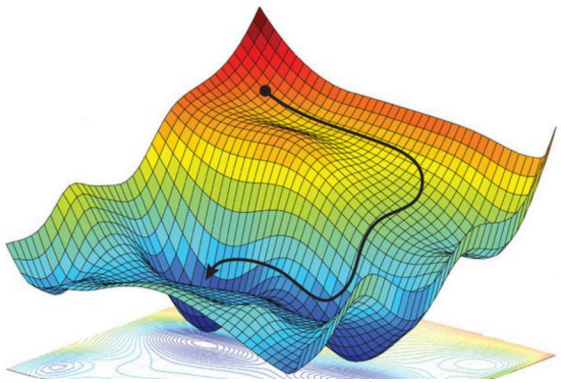


Figure 11: Learning rate comparison

A good analogy for the learning rate is teaching a child, where choosing the learning rate is akin to selecting the difficulty level of exercises. If the exercises are too challenging, the child may become overwhelmed and disengage. Conversely, overly simple exercises may hinder progress. Optimal learning occurs when the exercises gradually increase in difficulty, encouraging steady improvement without overwhelming the child. To better understand the learning rate's function, it's important to grasp the role of the optimizer. The optimizer is vital for training the model, as it regulates the size of the learning steps. Initially, the optimizer takes larger steps, gradually reducing them until finding the optimal learning rate, as depicted in the following images:



(a) Learning steps [8]

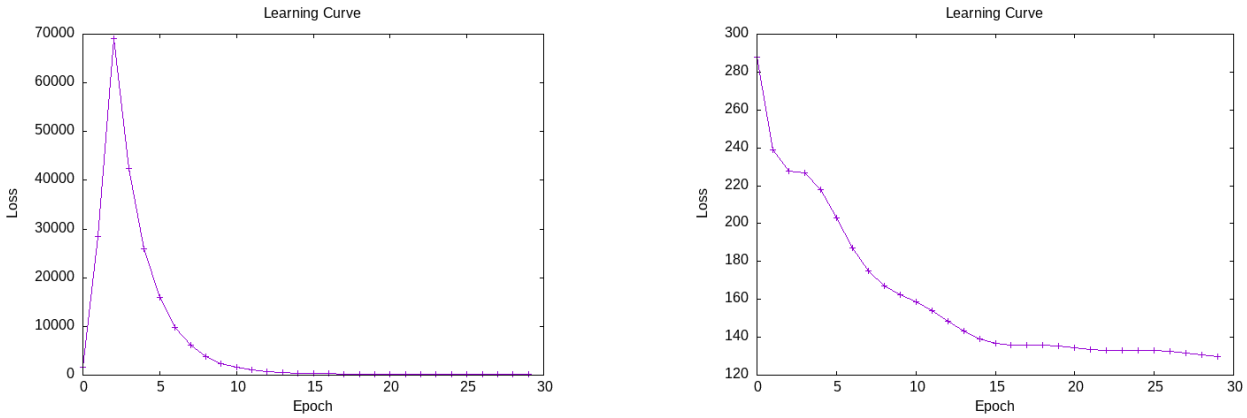


(b) Visual look at optimizer [10]

Figure 12: optimizer visualisation

The optimizer can be likened to a ball rolling down a slope, with the bottom representing the optimal point. Various implementations of optimizers exist, with the most common ones being Stochastic Gradient Descent (SGD), momentum, and Adam. Each optimizer combines different

concepts to achieve smoother and more effective learning curves. SGD is the simplest, utilizing only the gradient of the curve to determine if the point is at the lowest spot. However, it may stop prematurely if it encounters a point where the gradient equals zero, even if it's not the global minimum. Momentum enhances SGD by incorporating the concept of momentum, similar to a ball rolling downhill gaining speed. Adam, short for Adaptive Moment Estimation, is a sophisticated optimizer that combines elements of both SGD and momentum, resulting in improved performance. Here are two loss curves from my Titanic model, trained using SGD and Adam:



(a) SGD (b) Visual look at optimizer

Figure 13: Optimizer comparison

The optimal accuracy for this model reached around 80%. Comparing the loss curves, we observe that when using SGD, the loss initially spikes and then rapidly overlearns. In contrast, Adam's loss curve is smoother and converges to a better sweet spot within 30 epochs.

5.4 Classification models, Activation and Loss Functions

5.4.1 Classification Model Cifar10

After mastering the basics, I delved into more complex application cases, such as developing a classification model capable of identifying objects or animals in images. These types of problems require more than a simple neural network structure with just one input layer, one hidden layer, and one output layer. This is where MultiLayer Perceptron (MLP) models come into play. MLPs are characterized by having at least three different layers (input, hidden layer, and output), and the activation function of the hidden layer neurons must be non-linear. In my case, the optimal hyperparameters, balancing computational demand and performance, were (i=3072, h1=256,

h2=256, o=10). The input layer consisted of 3072 neurons due to the image size being 32x32 pixels, and the output layer had 10 neurons because there were 10 different classes to predict. Choosing the number of hidden layers and neurons in each layer is more challenging, as there isn't a definitive method for it. Generally, increasing the number of neurons and layers improves accuracy but also increases computational demands. Thus, finding the optimal hyperparameters involves tweaking parameters until the best configuration is achieved.

5.4.2 Activation and Loss Functions

During the design of my Cifar10 model, I learned that selecting the activation function for the output layer depends on the problem type and the desired data interpretation. For classifying images into ten different categories, I opted for the Softmax function as the activation function. Softmax not only provides probabilities for each class prediction but also enables insights into the model's certainty through tools like confusion matrices. Loss functions play a crucial role in guiding the model towards the correct answers. Among various options, the Mean Squared Error (MSE) loss function stands out for its effectiveness in highlighting model errors. With Softmax as the output layer activation function, the CrossEntropyLoss function is more suitable. This combination shapes the model's output as a list of probabilities for each class in the image, facilitating accurate classification. In general, the choice of activation function for the output layer strongly influences the selection of the loss function.

5.5 Data Analysis, Confusion Matrix

Now that I had a functional model, I needed to ensure it wasn't overfitting the data and monitor its performance during training to determine if it was training effectively. To accomplish this, I learned multiple techniques and best practices to identify such issues.

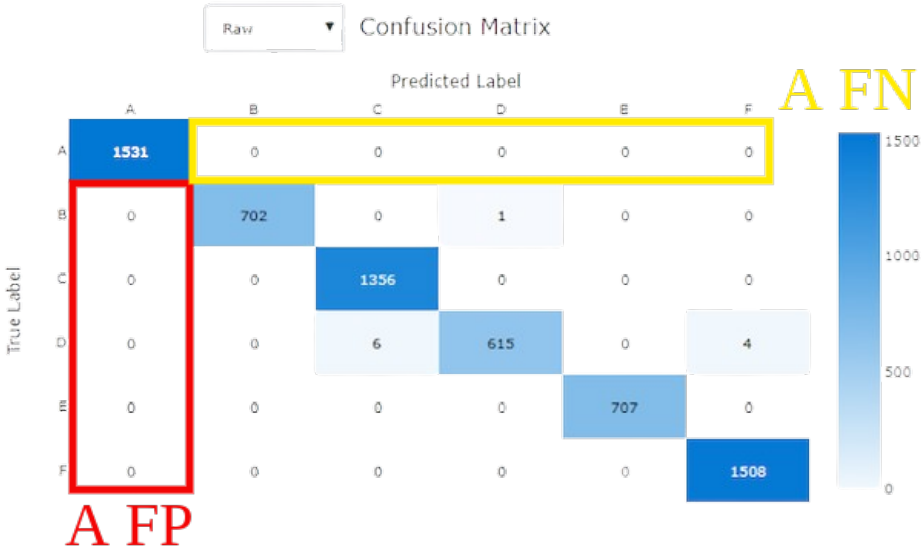


Figure 14: Confusion Matrix
[4]

Your explanation of the Confusion Matrix and the importance of F1 scores is clear and informative. Here's a refined version: The Confusion Matrix is a valuable tool for identifying areas where the model struggles. To interpret it, each column represents the predicted labels or classes, while each row represents the expected labels. From the Confusion Matrix, we can extract important information such as True Positives, False Negatives, and False Positives, which are essential for assessing the model's performance. Using these metrics, we can calculate various indices like the Accuracy of the model and F1 scores. While accuracy provides a general measure of model performance, it lacks granularity. For instance, if the model achieves 98% accuracy across all classes, we don't know where the 2% of misses occur. This is where F1 scores come in. They provide a more detailed assessment of model performance for each class, offering insights into areas of weakness.

5.6 Word to Vector

Having gained foundational knowledge about neural networks and data analysis, it was time to delve into Word2vec. Word to vector, commonly known as Word2vec, is a technique used to convert words into vectors. The objective is to represent each word as a vector so that operations can be performed on these vectors to determine semantic relationships between words. For example, by subtracting the vector for "woman" from "queen" and adding the vector for "man," we should arrive at a vector closely resembling that of "king." Word2vec is also valuable

for identifying synonyms and sentences with similar meanings, even when using entirely different words.

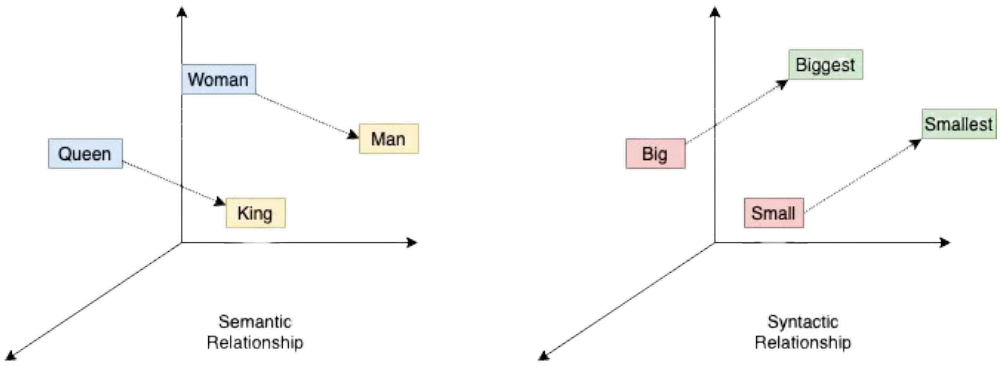


Figure 15: Word2vec visualisation
[2]

To create word vectors, I explored two algorithms: CBOW (Continuous Bag of Words) and skip-gram. Opting for CBOW, I found it to be more straightforward for training. The concept involves training a model to apply the chosen algorithm and selecting the weights of the layer between the last hidden layer and the output layer, known as embeddings. These embeddings can have multiple dimensions, allowing for nuanced word representations, albeit requiring more computational power. In designing the model, I decided to remove the biases of the neurons. This was achieved by implementing an MLPHyperparameters tool in Hasktorch-tools, enabling me to control bias inclusion in each layer. The rationale behind removing biases was to assign equal importance to each word, irrespective of its frequency in the training set. However, Bekki-sensei later explained that biases are crucial for determining the tonality of the text, as they represent word frequency. By analyzing biases, we could discern that the training set comprised Amazon reviews. Despite this insight, removing biases allowed the model to learn a more generalized vocabulary. Furthermore, refining the training text is essential for maximizing information extraction from each word. Techniques such as text normalization (e.g., converting all words to lowercase), removing non-alphabetic characters, and filtering out common particles (e.g., "is," "at," "the") are vital steps in this process.

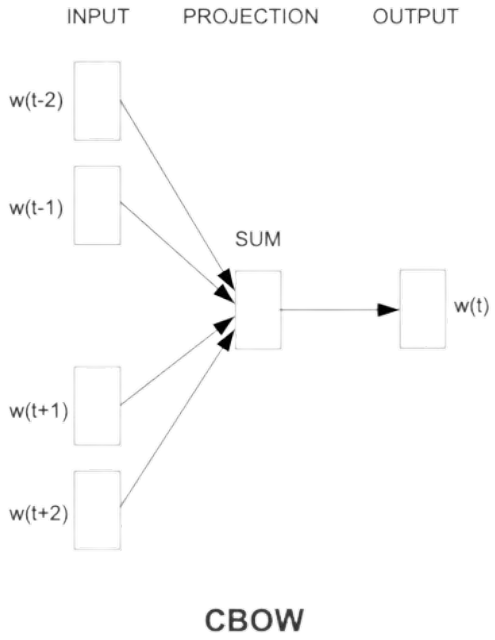


Figure 16: continuous bag of words
[2]

To illustrate the CBOW algorithm, let's consider an example. Given the sentence "Everyone at BekkiLab is friendly," and if we input the words "everyone," "at," "is," and "friendly" into the model, it should predict "BekkiLab" Now, here are two examples of inference using my implementation:

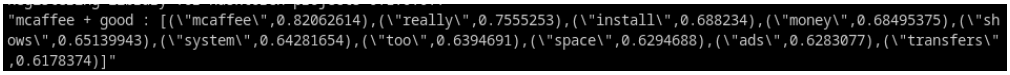


Figure 17: mcaffee + good

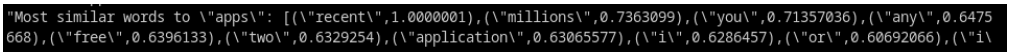


Figure 18: words similar to apps

An important observation I've made regarding this aspect is the necessity for more data. During implementation, I encountered issues, which I'll elaborate on shortly. I was only able to load ten thousand words from the dataset, which comprises one million Amazon reviews worth of text. I believe that having a larger dataset with a greater variety of samples would have resulted in more precise word vectors.

5.7 Recurrent Neural Networks

Finally, let's explore Recurrent Neural Networks (RNNs). Unlike traditional linear networks and MLPs, RNNs were designed to address the issue of memory limitation in classic neural networks. Traditional networks lack access to their previous outputs, meaning that each input is processed in isolation. RNNs resolve this limitation by providing the previous output as input to the next iteration, thereby offering greater context and enhancing their ability to learn sequential data.

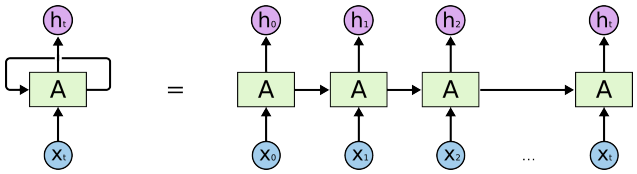


Figure 19: RNN
[6]

On the left, we have a schematic representation of an RNN, while on the right, we see its unrolled version. In the unrolled version, each instance of input, denoted as x, is processed sequentially by the layers, denoted as A, producing an output h at each iteration. One of the most challenging aspects of implementing an RNN is formatting the data to create a suitable training set.

5.8 Societal and environmental impacts

5.8.1 Impact of commuting

In order to go to the lab I had to take the subway for a distance of 6km twice a day, for 5 days a week, for 9 weeks in total I had to travel 432 km which is around 0.01 metric tons of CO2

5.8.2 Impact of the travel

I had to board on 3 planes to come to Tokyo, Mérignac BOD airport to Paris CGD airport to Seoul GMP airport and finally to Tokyo HND airport. The addition of the outbound plane and the return plane has a 2.87 metric tons of CO2 emission.

5.8.3 Impact of the computer equipment used

Tokyo's CO2 emission by kWh is 0.4615 kgCO2e/kWh 18.69075, as my notebook consumes 10 Watts and I use it around 9h a day per day of work it consumes 0.9 kWh, we can multiply 0.9 times 5 days, times 9 weeks. At the end of the internship my computer consumes around 40.5 kWh and multiplying by 0.4615 we have 18.69075 kgCO2e so around 0.02 metric tons of CO2 emission.

5.8.4 Conclusion

Summing up everything I have emitted a total 2.9 metric tons of CO2 for work. As we should aim for a maximum of 2 metric tons of CO2 per year, this is a pretty bad score. It is obvious that the plane is the major reason for this bad score and using public transportation was all I could do to mitigate it. All the calculations where done on https://www.carbonfootprint.com/calculator.aspx.

6 Conclusion

To conclude this report, I can't forget to say that the subject of the internship couldn't be a better one. I was already passionate about machine learning before the internship and Haskell interested me a lot by its lazy evaluation so it was a great surprise to know that I could fuse both things. Now I am more passionate then never about the subject and as I plan to make machine learning my career, I will keep on researching with the tools I used in this internship. The biggest bottleneck of the internship was the lack of computational power, as my computer wasn't suited to training neural networks, but it wasn't too big of a obstacle to learn machine learning, it just limited the scale of my projects.

References

[1] *About Ochanomizu*. Ochanomizu University. 2024. URL: https://www.ocha.ac.jp/en/introduction/index.html (visited on 04/15/2024).

[2] N. Birajdar. "Word2VEC Research paper explained - towards data science". In: *Medium* (Mar. 2022). URL: https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30.

[3] Wikipedia contributors. *Sigmoid function*. https://en.wikipedia.org/wiki/Sigmoid_function. Accessed: 2024-05-24. May 2024.

[4] Manash Goswami. *Evaluate AutoML experiment results - Azure Machine Learning*. Microsoft Learn. Aug. 2023. URL: https://learn.microsoft.com/en-us/azure/machine-learning/how-to-understand-automated-ml?view=azureml-api-2.

[5] K. E. Koech. "The Basics of Neural Networks (Neural Network Series) — Part 1". In: *Medium* (May 2022). URL: https://towardsdatascience.com/the-basics-of-neural-networks-neural-network-series-part-1-4419e343b2b.

[6] Chris Olah. *Understanding LSTM Networks*. Aug. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[7] R. Paolucci. *Linear regression v.s. neural networks - towards data science*. Accessed: 2024-05-29. Medium. Dec. 2021. URL: https://towardsdatascience.com/linear-regression-v-s-neural-networks-cd03b29386d4.

[8] Pushkar. *Understanding optimizers in deep learning: exploring different types*. Accessed: 2024-05-29. Medium. May 2023. URL: https://medium.com/codersarts/understanding-optimizers-in-deep-learning-exploring-different-types-88bcc44ff67e.

[9] Lavanya Shukla. *Designing your neural networks*. https://www.kdnuggets.com/2019/11/designing-neural-networks.html. Accessed: 2024-05-24. 2019.

[10] Tech-Ai-Math. *Optimization Techniques in Machine Learning (part 1)*. Accessed: 2024-05-29. Medium. Sept. 2023. URL: https://tech-ai-math.medium.com/optimization-techniques-in-machine-learning-8b4f7325295.

[11] TNG Technology Consulting GmbH. *MUNIHAC 2020: Austin Huang - Hasktorch: Differentiable Functional Programming in Haskell*. YouTube. Sept. 2020. URL: https://www.youtube.com/watch?v=Qu6RIO02m1U.

Glossary

C++ Programming language. 7

Haskell Functional programming language based on lambda calculus. 2, 7, 20

Hasktorch Haskell library based on Torch. 2, 5, 7, 9

inference The process of making predictions or decisions based on a model, often used in the context of machine learning models applying learned knowledge to new data. 7, 17

Libtorch C++ library for machine learning. 7

Linux Open-source operating system based on Unix. 7

MacOS Operating system developed by Apple Inc. for its Mac computers. 7

mathematical linguistics Field that combines linguistics, logic, philosophy, and natural language processing.. 2, 5

Python Programming language. 7

PyTorch Python library based on Torch. 7

sequential data Data that is ordered in a sequence, where the order of the data points is meaningful, such as time series data, sentences in a paragraph, or video frames. 18

Windows Operating system developed by Microsoft. 7

YAML A human-readable data serialization standard that can be used in conjunction with all programming languages and is often used for configuration files. 7