One Shot Learning Using Siamese Neural Network

Fawaz Sapa
Student
Seneca Polytechnic
Toronto,ON,Canada
fsapa@myseneca.ca

Tanishk Bisht
Student
Seneca Polytechnic
Toronto,ON,Canada
tbisht@myseneca.ca

Abstract

An important aspect of any machine learning model implementation is its ability to generalize which is a key aspect to tackle issues such as increase in development time and computational resources due to re-training of models as they were not built with the idea of generalization in mind. This paper further improves the capability of already capable one shot earning mechanism namely Siamese Network with adding features such as data augmentation, dropout regularization, and attention mechanism. We will be doing so by building upon the idea and CNN architecture given by the Siamese Neural Network for one-shot image recognition.

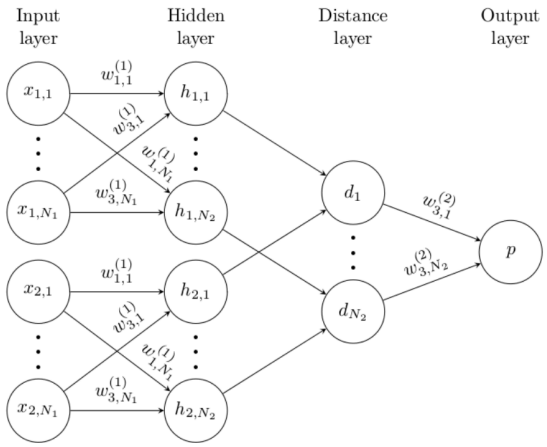
1.Introduction

The idea behind Siamese Network is humanizing the model i.e to train it in a way that humans are which is fascinating and we aren't just randomly creating layers with weights,biases and activation.What we found interesting is the thought process behind it.Just as the humans learn to classify and recognize things with a limited data or in this case information Siamese Network aims to achieve something similar.To be more technical,The focus is the ability of a model to classify data points correctly after being exposed to only a single example from each class during training.

2.Original Work

The original work was done based on the Omniglot dataset which contains examples from 50 alphabets ranging from well established international languages.The complete data set can be requested through brenden@cs.nyu.edu. Each character in Omniglot is a 105x105 binary-valued image which was drawn by hand on an online canvas.. The Omniglot dataset has a variety of different images from alphabets across the world.from Latin to Korean to lesser known alphabets. It also includes some fictitious character sets such as Aurek-Besh and Klingon (Figure 1).

2.1Overview of model



The paper utilized twin networks which are joined at their outputs. These networks were designed to process pairs of inputs and compute a similarity metric between them.This similarity was based upon L1 component-wise distance which is calculated between the output of the twin networks,a lambda layer calculates the L1 distance between them, and a dense layer with sigmoid activation predicts the probability that the pair belongs to the same class.The model used rectified linear units (ReLU) in the initial layers and sigmoidal units in the output layers.The paper stated to “initialized all network weights in the convolutional layers from a normal distribution with zero-mean and a standard deviation of 10−2 biases in the convolutional layers are initialized from a normal distribution with a mean of 0.5 and a standard deviation of 10^-2.The model was trained using a cross-entropy loss function and

optimization techniques like backpropagation and stochastic gradient descent were employed. L2 regularization is used to prevent overfitting.Models performance was based on accuracy in classifying pairs of images as similar or different based on the learned similarity metric.

2.2Code

We weren't able to reproduce the original code due to mismatch in system and code requirements.The other reason was the high demanding specs of the code.Due to time constraint and absence of CUDA led us to select a code which wasn't that resource demanding and then evaluate it.The code can be accessed from: https://github.com/sorenbouma/keras-oneshot.

3.Code Reproduction

Initially, we cloned the GitHub repository associated with the research paper into our VS Code environment. Following this, we installed the necessary libraries specified in the requirements.txt file provided in the repository. Subsequently, we cloned the Omniglot repository to acquire the dataset needed for our study.

In the subsequent step, we encountered an issue with unzipping the images from the Omniglot dataset. The load_data.py file contained a malfunctioning unzip command, which we rectified using PowerShell's Expand-Archive command as a substitute.

Further difficulties arose with library imports, specifically with the imread function. This problem was resolved by transitioning from scipy.misc to the imageio library for image reading functionalities.

Upon unzipping the Omniglot dataset, we attempted to compile the Jupyter Notebook (.ipynb) file in VS Code. This process was hindered by a ModuleNotFoundError for Keras. We addressed this by modifying the imports, replacing keras with tensorflow.keras.

Additional compatibility issues with TensorFlow were encountered; the version in use within VS Code was 2.16, along with Python 3.11. Given that the codebase was six years old, we attempted to create a separate environment using Python 3.7 and installed TensorFlow 2.2.0, but we were unable to execute the .ipynb file successfully. Consequently, we opted to use Google Colab for compilation. We transferred the image data to Google Drive, which enabled successful execution of the notebook.

4.Suggestions

The code does pretty well to replicate the source paper.There are still areas for improvement.Based on our resources we were able to create and implement some suggestions on the source code.We tried methods to improve the models ability to generalize more efficiently.

4.1 Methods used

The original paper contained data augmentation to improve model generalization ability but the source code we used did not contain data augmentation hence we decided to integrate it in the code so as to enhance our understanding in image manipulation.

Leveraging TensorFlow's ImageDataGenerator we included rotations, translations, zoom, flips, and slight color variations which would help the model to focus on the essential features of the input images instead of minor changes. We used rotation_range=10 to rotate the image by any angle from -10 to 10 degrees, width shift (width_shift_range=0.1) and height shift (height_shift_range=0.1) to shift the image horizontally and vertically by 10%. And used zoom_range=0.1 and shear_range=0.1 to randomly zoom inside the pictures and to use shear angle in the counterclockwise direction in degrees. horizontal_flip=True randomly flips half of the images horizontally.

Regularization: We utilized dropout layers (Dropout(0.3)) after MaxPooling layers in the network to randomly set 30% of the node activations of input units to 0 at each forward and backward pass, which helps prevent overfitting.Along with using L2 regularization(ridge regression) on the convolutional layers (kernel_regularizer=l2(2e-4))and dense layers(kernel_regularizer=l2(1e-3)) to limit the weights during optimization.

Attention Mechanism: While researching we found this fascinating way of attention mechanism which allows the network to focus on the most relevant parts of the image, which might improve its ability to distinguish between very similar categories in one-shot learning.

The reason it is so fascinating is because that's how human's minds work as well. We all have an attentive zone that we pay particular attention to.

We have defined a custom function simple_attention that applies an attention mechanism on the feature maps from the last convolutional layer.

A softmax layer is used to create a normalized attention map. The softmax ensures that the sum of weights in the attention map equals one, effectively allowing the model to decide which parts of the image are most important.

The original feature map is then element-wise multiplied by this attention map. This step scales each feature by its corresponding attention weight, emphasizing features that are deemed important and suppressing the less important ones.

4.2 Comparison

The ways our suggested methods is advantageous over the source code is due to its ability to further generalize and improve performance on unseen data which can help the model learn more generalized and robust features through Data augmentation and regularization techniques.Integration of attention mechanism allow the model to focus on the most relevant parts of an image, improving its ability to distinguish between similar categories.

The suggested model has its shares of disadvantages as well adding regularization through dropout layers and attention mechanism is bound to increase the models complexity therefore increasing the resources and runtime compared to the original code.When we thought of the method we also feared that the added complexity would maybe not increase its accuracy if the base model was already fine tuned to performing in the most optimum way.In other words there is a risk of over engineering.

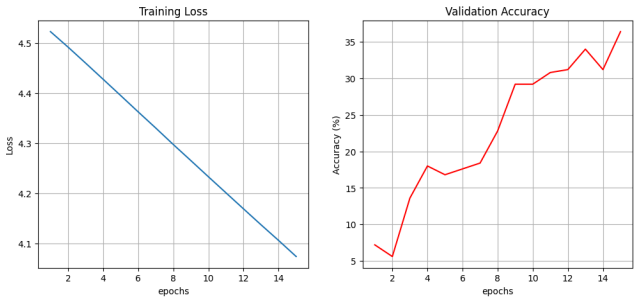
5.Performance Evaluation

After the modification in the code and by adding factors regularization, dropout rate, attention mechanism and data augmentation we were able to infer the improved performance of the modified code. Although these functionalities took more computational resources and time but were rewarded with increased performance in training loss and validation accuracy. Given is a table with a comparison of the original code and the modified code.

Original Code:

Epochs	Loss	Validation Accuracy (%)
1	4.521	7.2
5	4.40	16.8
10	4.23	29.2
15	4.07	36.4

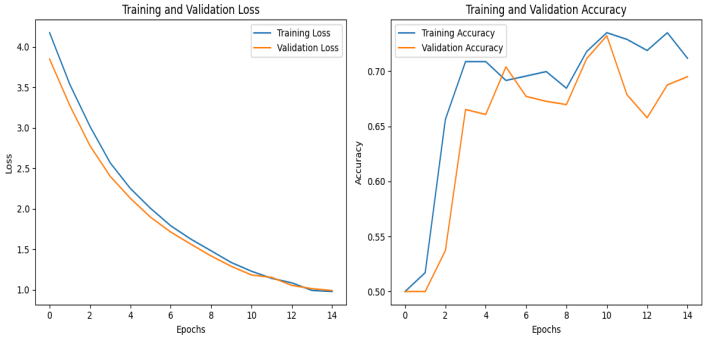
From the table given above of the original code performance we can infer a gradual decrease in loss over 15 epochs, which is expected as the model learns from the training data. But was not significant enough as compared to modified code performance. The original code starts with a very low validation accuracy that only modestly improves over time. Below is a graphical representation of the original code's performance:



Modified Code:

Epochs	Loss	Validation Accuracy (%)
1	4.175	50.0
5	2.5693	66.5
10	1.338	71.2
15	0.979	70.1

The modified code shows a much more significant decrease in loss, starting at a similar point but ending much lower after 15 epochs. This suggests that the modified model is learning more efficiently. The modified code starts at a 50% validation accuracy, which is significantly better, and improves steadily, reaching a peak of 71% at 10 epochs. It's interesting to note that there's a slight decrease at 15 epochs, suggesting possible overfitting or that the learning rate might need adjustment. Below is a graphical representation of the modified code's performance:



Critical Improvements

- Data Augmentation: Data augmentations is one of the reasons for the improved performance. Augmentation artificially increases the size and variability of the training dataset, which helps prevent overfitting and allows the model to generalize better to unseen data.

- Attention Mechanism: We used Lambda(simple_attention) function before flatten which applies a softmax function across the feature maps contributing to the improved performance.

- Dropout Rate: We applied a drop rate of Dropout(0.3) after the max-pooling layers within the CNN architecture. By dropping out different sets of neurons on different training passes, dropout forces the network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons helping in increasing the performance.

- Learning Rate: There might be changes in the learning rate or optimizer settings. We applied a learning rate of optimizer=Adam(0.00006) which led to improved training outcomes.

6.Conclusion

With the race for AI technology at its peak with massive investments being made.Finding large datasets while also maintaining the quality and regulatory concerns is challenging.This calls for methods such as one-shot learning aims to make accurate predictions with extremely limited data—just one example per class.

To sum up, specific modifications made to the original Siamese network have greatly enhanced its functionality. The network's discriminative power is increased by adding a simple_attention function, which instructs it to prioritize relevant features. By pushing the model to learn more robust and redundant representations, the Dropout layers reduce overfitting.A key factor in exposing the model to a variety of changes and improving its capacity to generalize is data augmentation. Together with a precisely calibrated learning rate, this results in a more methodical and consistent convergence during training, by the continuous decline in loss and increased accuracy of validation.

All these improvements—focus on critical features through attention, the increased generalizability with dropout and data augmentation, and the careful optimization with an adjusted learning rate—have made the Siamese network more effective and efficient for one-shot learning tasks, which is crucial for applications where data is scarce.

REFERENCES

(2019, February 21). *Keras-oneshot*. Github. https://github.com/sorenbouma/keras-oneshot/tree/master

(2015, July 9). *Siamese neural networks for one-shot image recognition*. Paperswithcode. https://paperswithcode.com/paper/siamese-neural-networks-for-one-shot-image

(2020, September 2). *A friendly introduction to Siamese Networks*. Towardsdatascience. https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942