Machine Learning: Developing an Algorithm That Can Read Your Hand Language

Victor Espidol and Adam Novak

ABSTRACT - **Machine Learning is a powerful tool that allows a system to learn and improve from iterations and runs without actually changing the code or the algorithm of that system. Essentially the system or program can access data and use it for itself to modify and tweak algorithms and patterns to allow the system to learn and develop itself.**

The goal of this project is to develop a machine learning pipeline that follows the general workflow of a supervised learning algorithm, which includes gathering data, feature generation, pre-processing of data, selecting a model for the algorithm, tune the model, and then evaluation of the model. The purpose of this project is to take sign language characters A-I and develop a machine learning pipeline that creates a model based on training data, and then use this model to evaluate itself on other test data (different images of the same sign language characters). Essentially the model is developed using given data, the model learns patterns for each sign language character, and then this model can be used for new, never seen data to evaluate its performance and accuracy for new data.

From experimentation with different models, it was determined that Convolutional Neural Networks (CNN) is a suitable model to determine sign language characters, as this is typically a strong performer for image processing. The machine learning pipeline using the CNN had an accuracy of over 97% for training data and for the denoted easy characters a score of over 93% and an average for the more difficult characters of 87%. There were other models that were successful in determining sign language characters too (Multi-layer Perceptron and K-NN), however the CNN model outperformed these by roughly 5-10%.

INTRODUCTION

Many advancements have been made in the computing world over the last decade, and one of those is Machine Learning. Machine Learning is a powerful tool that allows systems to take an abundance of data and make models that can be used to predict various things such as images, future events, etc.… The system can spot patterns and modify the patterns as they learn with more and more data to better predict new, unknown data. This concept allows machine to derive various knowledge from large amounts of data in order to predict new outcomes and as a result, this reduces the need for humans to take large amounts of data and analyze all these different outcomes, which allows for better predictive knowledge and performance in a more timely manner (Raschka, Mirjalili). As a result of the robust power of machine learning we have powerful tools such as spam filters for emails, scam alerts on phone calls, image processing and prediction for the medical field and beyond, stock market prediction, and self-driving cars.

There are three main different types of machine learning algorithms: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Supervised Learning is the method that involves using labeled data and direct feedback that predicts an outcome or future event (Raschka, Mirjalili). Unsupervised Learning is a method that involves data that has no labels or identification that has no feedback when tested on, and the model comes from finding the hidden structure in the data (Raschka, Mirjalili). Lastly, Reinforcement Learning involves using a decision process that learns from a series of actions and involves using a reward system (Raschka, Mirjalili). Overall, the method to stress is supervised learning as this is the method used for this project. Supervised Learning involves having training data tied to its label, which is fed into a machine learning algorithm (such as a multi-layer perceptron, or a simple linear regression) to develop a model. This model can then be used to predict the labels of new data given to determine its performance (Raschka, Mirjalili).

This project involves using the Supervised Learning machine learning algorithm to perform image processing and prediction. The purpose is to be able to take a set of images of different people signing the characters A-I and be able to produce a machine learning algorithm that can determine what character each image is representing. The training data is comprised of 1844 images total with each corresponding label of sign characters A-I. Our machine learning pipeline uses this training data to develop an algorithm or system that will be used to

test new image data using multiple different methods of creating an algorithm. Then using these various algorithms, multiple performance metrics are used to determine which algorithm is the most robust by testing the algorithm with the same training data and new test data. The next section will discuss how the machine learning pipeline was implemented and discuss which algorithms were decided upon.

IMPLEMENTATION

A typical machine learning pipeline involves data acquisition, feature generation, pre-processing of data, model selection, tuning of the model, and performance analysis. For the machine learning pipeline, the pre-processing of the data is the same for all the different models, however the models are what change. In this section, the machine learning algorithms of K-NN, Random Forests, Multi-layer perceptron, and Convolutional Neural Networks (CNN) will be discussed as these were the algorithms that tend to do very well with image processing and prediction. These are the classifiers/models used and tested upon to help our team determine which is best for classifying sign language characters.

The workflow of the supervised machine learning pipeline begins with data acquisition. The data acquired includes 1844 images of sign language characters ranging from character A to character I. Each image is comprised of 100X100X3 pixels. Thus, there are 10000 pixels all valued from 0 to 255 for red, green, and blue. The data was then pre-processed as this is important for the model. Our data was both normalized and standardized. Normalizing data means to scale all data values from 0 to 1 (Raschka, Mirjalili). So, for example, the images are of RGB data ranging from 0 to 255, thus, the normalized data points would compress the original data value by 255 to get the RGB data to range from 0 to 1. Then, our data was also standardized which modifies the data so that the mean is 0. Lastly, the data was compressed to gray scale. This reduces the dimensionality of the image from 100X100X3 to 100X100 and averages out the RGB values to produce a value for just a gray scale image. Each of these processed packets of data (normalized image data, standardized data, and gray scale data) can then be used for various machine learning algorithms to produce a model, that will then be used to predict new data. Next, the four different models used will be discussed as each of these were used to determine which is best for sign language image classifying.

The first model used for our machine learning pipeline is the K-Nearest Neighbors (K-NN) classifier. This classifier compares a test point to the k nearest training data points and then guesses the outcome value based on the desired ouput values of the k nearest training points (Silva, Lecture 11). For this algorithm, the dissimilarity measure used is Euclidean Distance, which is essentially absolute distance between points and there was no weight appended to each point. Thus, if there is a point in which 10 neighbors belonging to class 1 and 2 neighbors belonging to class 2, the point would be identified as class 1. This model was trained by varying the number of neighbors to look at (1-10) and the accuracy was then determined by the trained model with different neighbors. The results are discussed in the experiments section.

The next model used is Random Forests in the machine learning pipeline. Random forests is an ensemble learning method for classification. This model constructs many decision trees during training and then outputs the class of the individual trees (Silva, Lecture 12). These are useful as they tend to reduce the effects of overfitting done on just one decision tree (Silva, Lecture 12). For our pipeline we simply just changed the number of estimators (number of decision trees created) and altered the value from 1, 5 ,10, 20, 50, 100, and 500. Then once the model was created, the test data was tested to determine accuracy. The results are discussed in the experiments section.

The next model tested was a Multilayer Perceptron. This is essentially a feedforward artificial neural network involving multiple layers of perceptrons with activation. This consists of at least three layers of nodes, including the input layer, at least one hidden layer, and an output layer (Silva, Lecture 17). Each of these nodes is a neuron that uses a nonlinear activation function. Essentially, there is input at the input layer and doing calculations through the hidden layers and output layer, a result is yielded at the end. Then using backpropagation, the backward phase computes the error signal, meaning it propagates the error backwards through the network starting at the output, which can then be used to change the weights of the neurons to get the actual result closer to the yielded result (Silva, Lecture 17). For our project, the activation tanh and logistic were tested along with the sgd and adam solver, in which logistic and sgd yielded the best results. The results are discussed in the experiment section.

The last model tested was the convolutional neural network. This model has been tested and has become very popular for image recognition, object detection, segmentation, and many other tasks. Deep convolutional neural networks construct a feature hierarchy by combining low level features to create a high-level feature (Raschka, Mirjalili). For example, low-level features such as blobs or edges are

extracted from layers which are combined to create high level features such as in this case: sign language characters. The CNN computes feature maps from an input image essentially, in which it creates a patch of pixels in the image which is known as a local receptive field (Raschka, Mirjalili). An example of this is seen in the figure below. The image below shows how a patch of pixels is extracted to create a low-level feature. CNNs tend to perform very well because they are sparsely connected, meaning a single element of the feature map is connected to only a small number of pixels, versus trying to connect to the whole image as it is for MLPs (Raschka, Mirjalili). In addition, another reason CNNs do well is because of parameter sharing, in which the same weights are used for different patches of pixels of the image (Raschka, Mirjalili). As a result of these two benefits, CNNs tend to do very well at image processing, especially when comparing to MLPs, as this tends to reduce overfitting as it essentially results in nearby pixels being more relevant to each other rather than ones far away and reduces the number of weights in the network. (Raschka, Mirjalili).

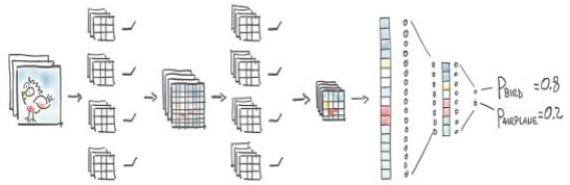

Figure 1: General outline of CNN

For our convolutional neural network, we created four different CNN models to test and compare. In each test we did an 80/20 train test split of the given dataset, used the Stochastic Gradient Descent optimizer, used the Cross-Entropy Loss criterion for our loss function, a batch size of 64, and an epoch from 50 to 100.

The first model uses the tanh activation function, max pooling and applies a linear transformation to the data. The function Conv2d is used which creates a convolution kernel that convolves with the input layer to produce tensors for output. The tanh activation function used has similar behavior to the logistic sigmoid, however the tanh ranges from -1 to 1 and is more ‘s’ shaped which allows for a strongly negative mapping for negative inputs whereas zero inputs are more closely mapped as zero (Sharma). After convolution and max pooling is done, a linear transformation is done twice to reduce the number of nodes to represent the 9 class labels. The model was then tested, and the results were, 97% training accuracy and 82% Test accuracy

The second model is essentially the same as the first however, batch normalization and a dropout zone was added to reduce overfitting and improve the accuracy. Batch normalization was used to reduce the amount by what the hidden unit values shift, which also reduces training time. Dropout is a method of regularization that approximates training a large number of neural networks with different architectures in parallel. This means that during training, some elements of the input tensor are randomly zeroed with a probability (set to 50%) using samples from a Bernoulli distribution.[1] This results in each channel being zeroed out independently on forward calls, which all is effective for regularization and preventing co-adaptation of neurons.[2] This model had the best results of the 4 CNN models with results, 100% training accuracy and 90% test accuracy.

The third and fourth models were the same as the first two models, with the exception that ReLU was used as the activation function instead. ReLU is very common to convolutional neural networks and for this function, f(x) is 0 when x is less than 0, and when x is greater than or equal to 0, f(x) equals x. To our surprise the change in activation function made the training and test accuracy fall below 80%. Therefore, the third and fourth models were no longer needed because the second model was the best.

EXPERIMENTS

Overall, all four algorithms proved powerful for image processing and prediction for this sign language experiment. For the performance evaluation, our data was split 80/20, meaning 80% of the given data was used for training and the other 20% was used for testing and this includes all characters A-I. The results of this setup is shown below in Figure 2.

Each algorithm did an excellent job at classifying the sign language characters with all classifiers having a training accuracy higher than 90%. However, the real proof of a working model is to validate it with real data or a test set. In these experiments we found that the CNN proved to be the best with a training accuracy of 100% and a test accuracy of 90%.

[1] Pytorch - https://pytorch.org/docs/stable/nn

[2] Pytorch - https://pytorch.org/docs/stable/nn

Overall, based on this data alone it appears that CNN is more optimal at image processing and prediction for our test purpose. In addition, it is important to note that the other three algorithms could possibly have achieved better accuracies, however, this may cause overfitting and actually decrease testing accuracy depending on how fine-tuned the parameters are.

Classifier	Training Accuracy	Test Accuracy
K-NN	100%	87.5%
Random Forests	100%	85%
MLP	98%	81%
CNN	97%	90%

Figure 2: Summary of Results for Characters A-I

Furthermore, additional testing was done on the "easy" characters, which comprise of sign language characters A-D. In this experiment we split the dataset into to only consist of the characters A-D and did an 80/20 train test split again. Based on the figure below (Figure 3), each of the four classifiers had excellent results for the training data in which the scores were all at least in 99% in training accuracy.

However, once again CNN outperformed the other classifiers as this classifier had a 98% test accuracy. The other algorithms also performed decently with their test accuracies near 90%, however CNN proved better in both cases, which is both believable and understandable, since CNN is a very powerful machine learning algorithm for image processing.

Classifier	Training Accuracy	Test Accuracy
K-NN	100%	92%
Random Forests	100%	89%
MLP	99%	88%
CNN	100%	98%

Figure 3: Summary of Results for Characters A-D

CONCLUSION

Based on the data collected from this machine learning pipeline experiment, it can be concluded that CNN is the best algorithm for image processing and prediction for sign language characters. This deep learning algorithm has already been noted as being an excellent classifier for images, and the data collected from this experiment has backed that claim up. For both test cases when testing the easy case versus normal case, the accuracy measures were the best for this classifier versus the other classifiers used. The accuracy score for the easy and normal test case of 98% and 90% respectively exceeded expectations and the accuracies for each other test. This algorithm is extremely useful as it reduces overfitting which yields high accuracy scores for both training and test data. However, the only downside to this algorithm is that it tends to take much more time to compute than each of the other algorithms.

REFERENCES

Raschka, Sebastian, and Vahid Mirjalili. *Python Machine Learning, Second Edition*. Packt Publishing, 2017.

Sharma, Sagar. "Activation Functions in Neural Networks." *Medium*, Towards Data Science, 14 Feb. 2019, towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

Silva, Catia. "Lecture 11 - K-Nearest Neighbors." 2020.

Silva, Catia. "Lecture 12 – Decision Trees & Random Forests." 2020.

Silva, Catia. "Lecture 17 – Multi-Layer Perceptron & Backpropagation." 2020.