Comparison of Machine Learning Algorithms for Video Classification on Youtube-8M Dataset

G3

1 Introduction

With the rapid development of data-driven computational algorithms and the growing trove of computer vision methodologies, there are now ample opportunities to obtain and analyze the high-dimensional data from the digital media. This project is motivated and inspired by the Google Cloud and YouTube-8M Video Understanding Challenge which invites contestants to build multi-label classification models that can allocate labels to each video from the YouTube-8M dataset: one of the most comprehensive large-scale video-based datasets that consists of pre-computed visual and audio features from over 7 Million video URLs. The total number of classes from the original dataset is 4716 and some videos can have multiple labels, the variability and the unregulated nature has made the classification task challengeable.

Due to the complexity of large-scale as well as the multi-label nature, only a subset with 10 most popular categories of videos is selected then used within our project scope. In addition, each video is selected only corresponds to one label to eliminate the multi-label nature. The objective of the project is to develop a classifier that assigns the class label based on given features (both visual and audio) of the video using a subset of the Google's recently released YouTube-8M dataset. In the subset, there are 46171 instances (videos) in total. For each instance, there are 1024 visual features, 128 audio features and one label associated with the video. There are a total number of 10 labels across all instances. A more detailed table of the subset is shown below:

Category	Game	Vehicle	Concert	Food	Animal
Frequency	13068	10996	9659	3564	2736
Category	Football	Mobile Phone	Toy	Outdoor Recreation	Nature
Frequency	1768	1527	1071	895	887

Table 1: Label frequencies of the YouTube-8M 1/50 subset

The YouTube-8M dataset is developed through a state-of-the-art process that transforms the video level features to numeric values. In order to keep the authenticity of the original video features, there will be no pre-processing on this dataset. To accommodate the multi-class nature, the one-vs-all approach is applied where we compare individual class with the rest and then assign the class with the highest output value to each instance. In addition, with the fact that this dataset has rather low interpretability, the main focus of the project would be on modeling and performance comparison.

2 Methods

2.1 TensorFlow

TensorFlow is an open-source machine learning library for numerical computation developed by Google and also actively used in a variety of current products such as Gmail and search engine. Tensors represent n-dimensional data arrays in this tool, vectors and matrices can generally be considered as first and

second order tensors. There are two reasons that TensorFlow has been used in this project: 1. It strongly supports deep neural networks as well as many other machine learning algorithms; 2. It has great flexibility, high efficiency and scalability among many algorithms especially on large-scale datasets.

2.2 Mini-batch Gradient Descent

Mini-batch Gradient Descent

One advanced feature of TensorFlow library is its built-in functionality that enables computation on a gradient of a function with regard to a set of inputs. This algorithm is also called Gradient Descent, which is essentially an optimization method to compute the coefficients of a function in order to minimize the loss function. With large-scale dataset, it is often not efficient to run iterations of the gradient descent for each instance (Stochastic Gradient Descent) which takes very long time to converge. On the other hand, computing the gradient using the whole training set (Batch Gradient Descent) can be great since all gradients are calculated in one run, however, it would be computationally costly for our large-scale data. Hence we use Mini-batch Gradient Descent, in which we use a small set of samples for every iteration, which is essentially an option in-between Batch Gradient Descent and Stochastic Gradient Descent. A simple loop pseudocode can explain how it works:

for i = 1 to num_of_iterations:

data_batch = sample_training_data(data, batch_size)
weights_gradient = evaluate_gradient (loss_function, data_batch, weights)
*weights = weights - learning_rate * weights_gradient*

The main advantage of using Mini-batch is that it enhances the computation efficiency by performing partially parallel computations to update the model. The parameter estimate updates would also witness a reduction in its variance which would stabilize the updates convergence. The main disadvantage associated with this method is that it involves an optimization on the size of the mini-batch. In our project, multiple sizes of the mini-batch would be selected and compared to propose the ideal batch size for the classification.

Adaptive Learning Rate

When applying the gradient descent to minimize the loss function, it is common to multiply the gradient by a coefficient that is essentially the Learning Rate. The value of the Learning Rate should not be too small in which case it would slow down the convergence, neither should it be too large which could lead to unstable and non-converging loss path. In this project, we adopt a process called Learning Rate Decay which performs an exponential decay on the Learning Rate. This process would decrease the learning rate while training the model and a Global Step value needs to be defined in order to obtain the decayed learning rate based on the following equation:

*decayed_learning_rate = learning_rate * decay_rate ^ (global_step / decay_steps)*

2.3 Logistic Regression

Multinomial Logistic Regression

Logistic Regression can be viewed as a ‘Linear Regression’ between the input variables and classes. The basic function that Logistic Regression model is constructed based on is:

$$P(y_i = 1 \mid x_i, w) = sigm(w_0 + w_i \cdot x_i)$$

In this equation, x_i represents the video feature vector and $y_i = 1$ or 0 represents whether it is classified as one particular label or not. The $sigm()$ function refers to the sigmoid function and it has been defined as:

$$sigm(x) = \frac{e^x}{e^x + 1}$$

In addition, the threshold value t has been predefined to form the decision rule of the classification for each label:

$$\hat{y}(x) = 1 \Leftrightarrow p(y = 1 | x > t)$$

The loss function is the cross entropy loss which is defined as below:

$$L(f(\hat{x}), y) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \ln f(x_i) + (1 - y_i) \cdot \ln(1 - f(x_i)))$$

Here, X is the set of inputs and Y represent the corresponding labels. The $f(x)$ denotes the output of the logistic regression model given input x_i and each of the y_i is binary.

L2 Regularization

When training the model, we have incorporated the regularization concept to prevent overfitting. The regularization based on L2 norm loss (also known as ridge regression) has been introduced in the classification model, this L2 penalized model's new loss function can be described as:

$$L'(f(\hat{x}), y) = L(f(\hat{x}), y) + \lambda\sqrt{\sum_{i=1}^{n} \| w_i \|_2^2}$$

Where we have now introduced penalties on the weights w_i, λ is a tuning parameter which we select its value based on the performance on the validation set and the value of λ was finalized to be 0.001.

2.4 Support Vector Machines

Support Vector Machine (SVM) can be used as a classifier that performs classification by finding the best hyperplane as decision boundaries. One main advantage of the SVM is that it implements instinctive complexity control to reduce overfitting. The key function of SVM is to maximize the margin around the hyperplane and this function is defined by a subset of all samples. A simple linear support vector classifier (SVC) can be built for a non-separable case via the following equations:

$$\underset{\beta_0, \beta_1, \ldots, \beta_p, \varepsilon_1, \ldots, \varepsilon_n}{\text{maximize}} \; M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i),$$

$$\varepsilon_i \geq 0, \; \sum_{i=1}^{n} \varepsilon_i \leq C$$

Where M is the width of the margin that is being maximized and the classification result is based on the sign of $f(x^*)$, C is a tuning parameter and $\varepsilon_{1,2,\ldots,i}$ are slack variables. Decision function $f(x^*)$ is defined as:

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \ldots + \beta_p x_p^*$$

The loss function for the SVC is hinge loss, it is also sometimes called the max-margin loss with respect to the optimization function, it is defined as:

$$L(f(\hat{x}), y) = \max(0, 1 - y \cdot f(\hat{x}))$$

Support Vector Machine is an extension of the SVC with kernels in order to expand the feature space, a simple SVM with Linear Kernel will have following function:

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

In addition to the Linear Kernel, the Gaussian Kernel has also been selected and experimented where now the Kernel function has been replaced as:

$$K(x_i, x_{i'}) = \exp(-\gamma \| x_i - x_{i'} \|^2)$$

Note that the gamma is another tuning parameter that is selected based on its performance on the validation set and the value of gamma was finalized to be 0.05. Both SVM models have also adopted L2 regularization, and the values of λ were selected based on the validation set. λ is 50.0 for Linear Kernel and 500.0 for Gaussian Kernel.

2.5 Artificial Neural Network

The figure below depicts the architecture of the Artificial Neural Network model built for the video-level feature classification, which corresponds to our model structure *Data Input [1024+128] – ReLU [300] – ReLU [150] – ReLU [60] – ReLU [30] - Softmax Output [10]*:

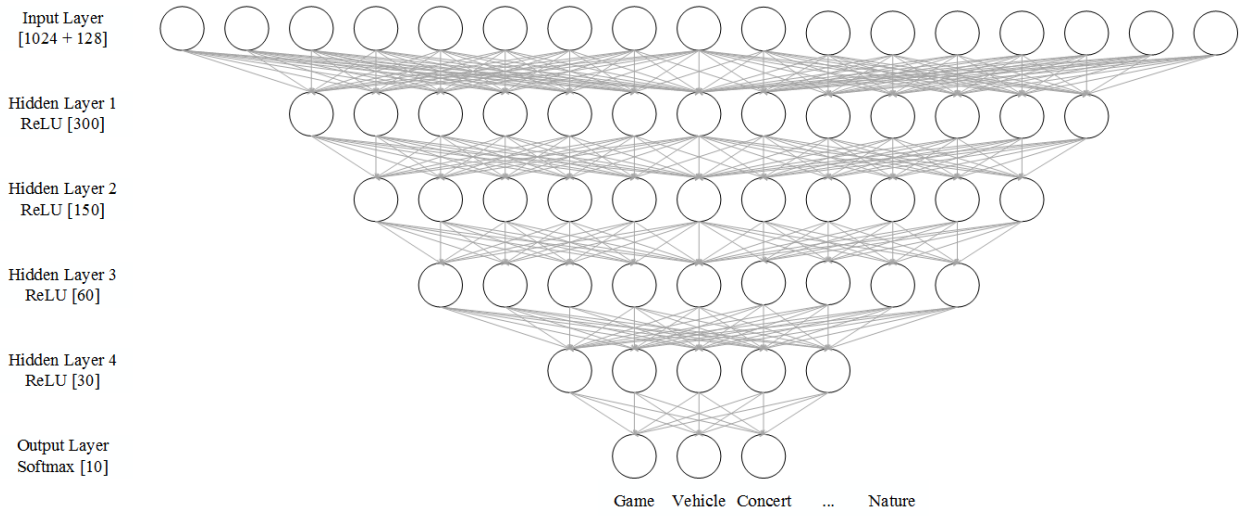


Figure 1: The architecture of the Artificial Neural Network

Hidden Layer

The Artificial Neural Network (ANN) is a nonlinear method that's analogous to the nature of human brains, with a fixed number of layers and neurons. One standard ANN model has an input layer, an output layer and a hidden layer. The input layer takes all the input values while the output layer generates the classification result. In the main essence, the hidden layer models nonlinear relationships by intervening between the input and output neurons. The first step in forming a hidden layer is to construct hidden features based on linear combinations of the input data. In our case, the pre-activation form for one instance is obtained as $WX + b$, where X is the input vector, W represents the weight vector and b is the bias vector for one instance.

Activation function: Sigmoid vs. ReLU

In ANN, each neuron would then require a nonlinear activation function and two most common functions are sigmoid and ReLU (Rectified Linear Unit). According to Krizhevsky et al, it was discovered that compared to the sigmoid function, ReLU could speed up the convergence and it's computationally inexpensive. Hence the ReLU function has been selected as the activation function for this project and it can simply be written as:

$$f(x) = \max(0, x)$$

Output function: Softmax

The outputs of hidden layers are then used as the input to the softmax function which generates the classification result:

$$Y_j = \text{softmax}(W_j X + b) = \frac{e^{W_j X + b}}{\sum_{k=1}^{K} e^{W_k X + b}}$$

The softmax function ensures the function output to be between 0 and 1, and the values could be interpreted as probabilities. The equation above is the basis for the neural network classifier in our model and the W and b matrices are obtained through the training process.

Loss function

Loss function here is the same as the cross entropy loss introduced in Multinomial Logistic Regression.

Backpropagation

In order to perform the gradient descent optimization within the multi-layer structure of our model, we compute the partial derivatives of the loss function $\partial L / \partial W$ and $\partial L / \partial b$ with respect to any weight w and bias b for in the neural network. With the fact that the loss would transit between multiple layers, an algorithm called Backpropagation has been adopted naturally. Let i^k and o^k denote the input and output vectors of the kth layer, W^k and b^k denote the transition matrix and the bias of the kth layer. Based on the ReLU function:

$$i^k = W^k o^{k-1} + b^k, \; o^k = \max(0, i^k)$$

Then suppose we have $X = o^0$ as the network initial input and $T = o^n$ as the predicted output with n layers, we can use the following equations to compute the partial derivatives:

$$\frac{\partial L}{\partial o^{k-1}} = W^k \frac{\partial L}{\partial i^k}$$

$$\frac{\partial L}{\partial i^k} = \frac{\partial L}{\partial o^k} \circ I[o^k > 0]$$

where $\circ$ means Hadamard product (element-wise). Since $\frac{\partial i^k}{\partial b^k} = I$ and we can use chain rule to derive the results:

$$\frac{\partial L}{\partial W^k} = \frac{\partial L}{\partial i^k}(o^{k-1})^T$$
$$\frac{\partial L}{\partial b^k} = \frac{\partial L}{\partial i^k}$$

Regularization: Dropout

Dropout is a technique to prevent overfitting problem of deep neural networks. The main objective of this technique is to randomly leave out neurons from the neural network when training the model and keep the networks "thinned". Instead of just evaluating network with a full set of neurons, the testing process would now requires averaging the classifications of multiple sets of thinned networks. Srivastava et al has found that this method can reduce overfitting for the ANN more significantly than other regularization methods.

2.6 Validation Process

With the constraint on the computing capability on the whole dataset, we move forward with multiple sample proportions of the subset (1/100, 1/50 and 1/25) as our inputs. For each subset, we randomly split it into 3 portions with 80% as the training set, 10% as the validation set and 10% as the testing set. Multiple algorithms would be applied and fitted on the training set and then validated, tested and assessed with respect to other methods.

3 Results

Table 2 displays the model performances comparison between multiple algorithms and we can conclude that ANN and linear SVC outperform other methods in terms of the classification accuracy. Logistic Regression is another satisfactory approach backboned by an accuracy of 91.8% on the testing set and SVM with Gaussian Kernel proves to be the least accurate model amongst all. From Table 3 we can see that the model performances show little sensitivity with regards to the sample sizes. However, the accuracy on the testing set does have a slight increasing pattern as the size of the mini-batch enlarges.

Accuracy (%)	Training Set	Validation Set	Testing Set
Logistic Regression	95.0	92.3	91.8
Linear SVC	94.9	92.6	92.4
SVM with Linear Kernel	91.4	87.6	87.1
SVM with Gaussian Kernel	75.2	72.1	71.7
Artificial Neural Network	98.0	92.7	92.6

Table 2: Model performance comparison with batch size = 256 and sample proportion = 1/50

Testing Accuracy (%)	1/100 of total data used	1/50 of total data used	1/25 of total data used
Mini-Batch size =128	91.2	91.2	91.5
Mini-Batch size = 256	91.2	91.8	91.9
Mini-Batch size =512	91.5	92.1	92.4

Table 3: Testing Accuracy of Logistic Regression with different batch sizes and sample proportions

4 Discussion

Overall, most of the models hold classification accuracies around 90%. Both Logistic Regression and Linear SVC are recommended with their simplicities and stabilities. One potential reason behind this could be the similarity between their loss functions, which has consequently lead to parallel loss minimization result. Due to the fact that a simple SVC only uses a subset of the training data to build the optimization function, it would be less sensitive to the sample size and more efficient and robust compared to the Logistic Regression.

In order to capture the nonlinearity amongst the features, both Linear Kernel and Gaussian Kernel have been tuned and then applied. However, both methods turn out to be less accurate and we can also observe that the model performance difference between the training set and the testing set is not significant, which indicates very small chance of overfitting issues. Under this circumstance, it can be inferred that the enlarged feature space from the Kernel functions is not useful for this specific task.

To further explore the nonlinearity of the features, Artificial Neural Network has been used and proved to be a viable option with its high classification accuracy. In addition, the network model with ReLU activation function trained also uses the shortest amount of time to converge. This is due to the simplicity of ReLU function when computing its gradient for optimization, which accelerates the convergence, especially compared to sigmoid activation function (see appendix).

After fine-tuning the parameters of training, the reason why a larger batch size could improve the performance is that its reduction on the variance of gradient updates by averaging the whole batch. In terms of the sample size, a large sample size was expected to be more powerful with more information loaded into the model. However, with the nature of this extreme large-scale dataset and undemanding classification task, the different proportions tested end up having very similar classification performance.

In order to further demonstrate a comprehensive analysis on this classification task, we recommend several other methods to be experimented and compared. For instance, the tree-based approaches and K-nearest neighbors could be applied. In terms of the kernel function applications, some further validation on the tuning parameters shall be implemented. It is also suggested to adopt other kernel functions that might be more suitable for this problem. Finally, it would be interesting to include more classes of videos and to perform classification towards whole YouTube-8M dataset using multiple executors and parallel TensorFlow.

5 References

[] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., & Vijayanarasimhan, S. (2016). Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.
Youtube-8m

[] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. Savannah, Georgia, USA.
Tensorflow

[] Aurélien Géron. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol, CA: O'Reilly Media, Inc.
Mini-batch gradient descent

[] Hastie, T. J., R. Tibshirani, and J. Friedman. (2009). *The elements of statistical learning: data mining, inference and prediction.* Second edition. Springer-Verlag, New York, New York, USA.
SVM

[] Moreira, M., & Fiesler, E. (1995). *Neural networks with adaptive learning rate and momentum terms* (No. EPFL-REPORT-82307). Idiap.
Learning rate

[] Fei-Fei, Li., Justin, Johnson., & Serena, Yeung. (2017). *CS231n: Convolutional Neural Networks for Visual Recognition*. Retrieved Mar 22, 2017, from Stanford University's CS231n Course Notes Web site: http://cs231n.github.io
minibatch gradient descent pseudo code
backpropagation

[] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
Relu,

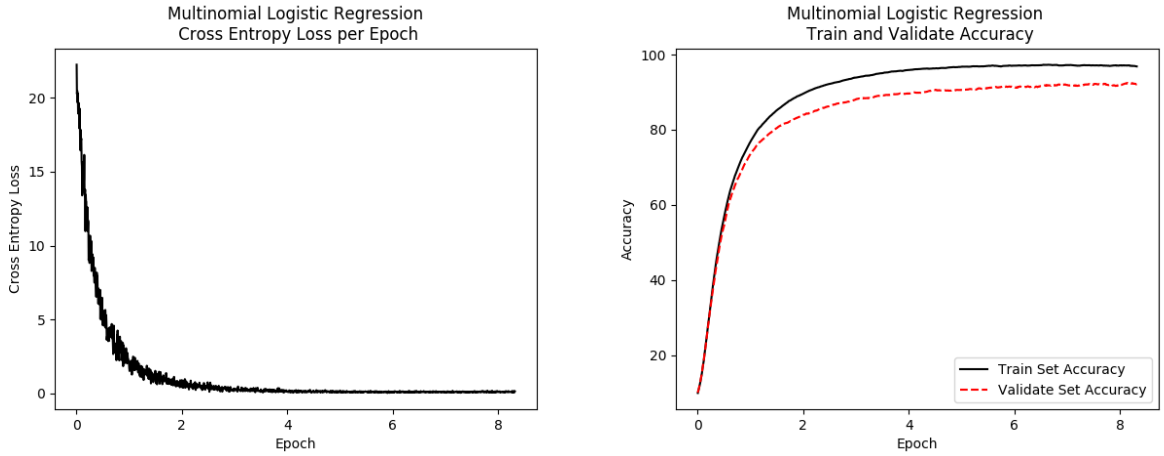
[] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929-1958.
dropout

6. Appendix

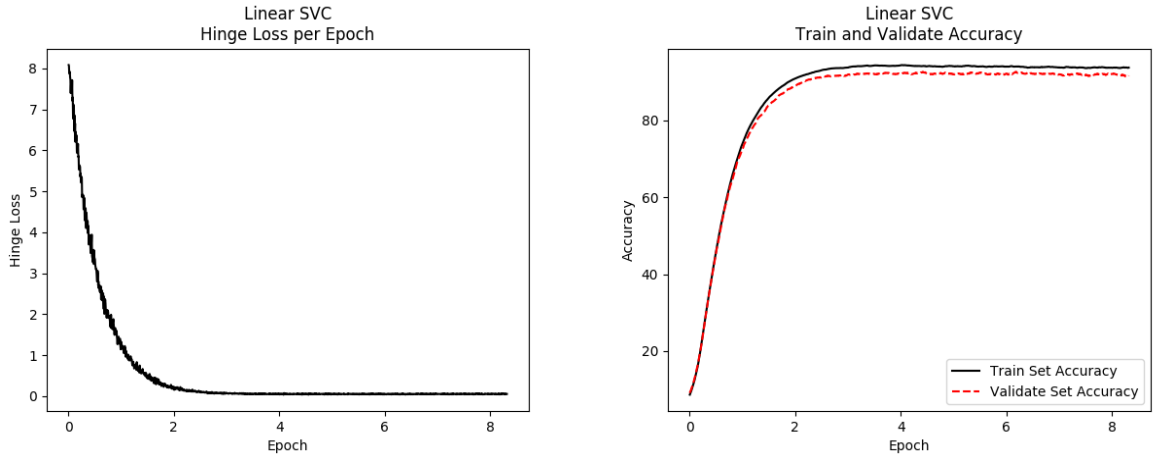
Code:

Please refer to the GitHub repository: https://github.com/noi10/CS6140_Project
logisticRegression.py
linearSVC.py
linearKernel.py
gaussianKernel.py
neuralNetwork.py

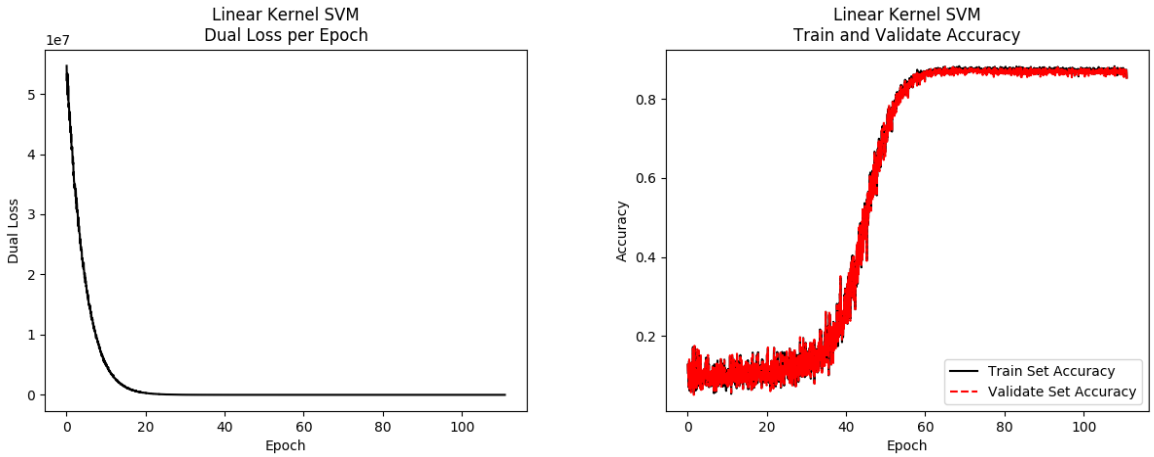
Figure:



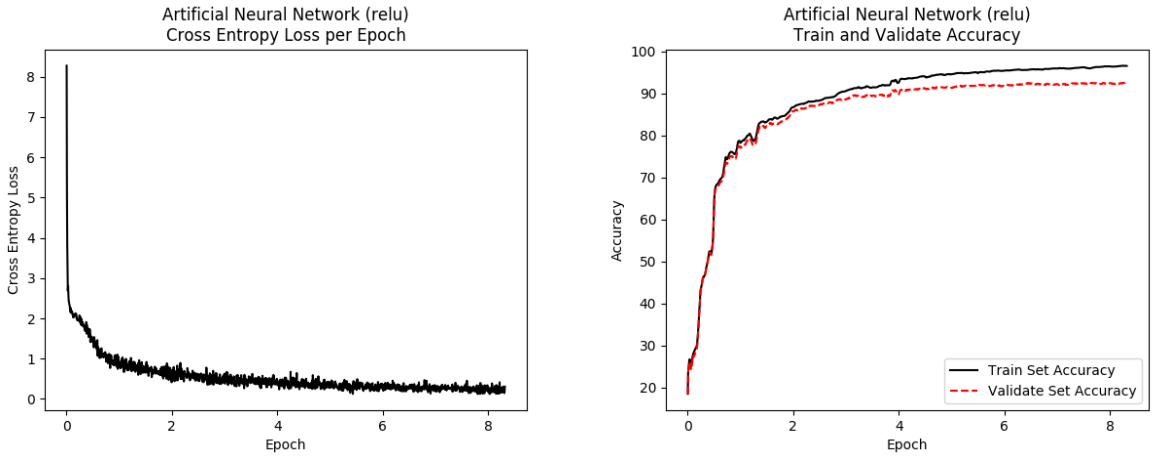
Appendix Figure 1: Loss and accuracy plots of Multinomial Logistic Regression



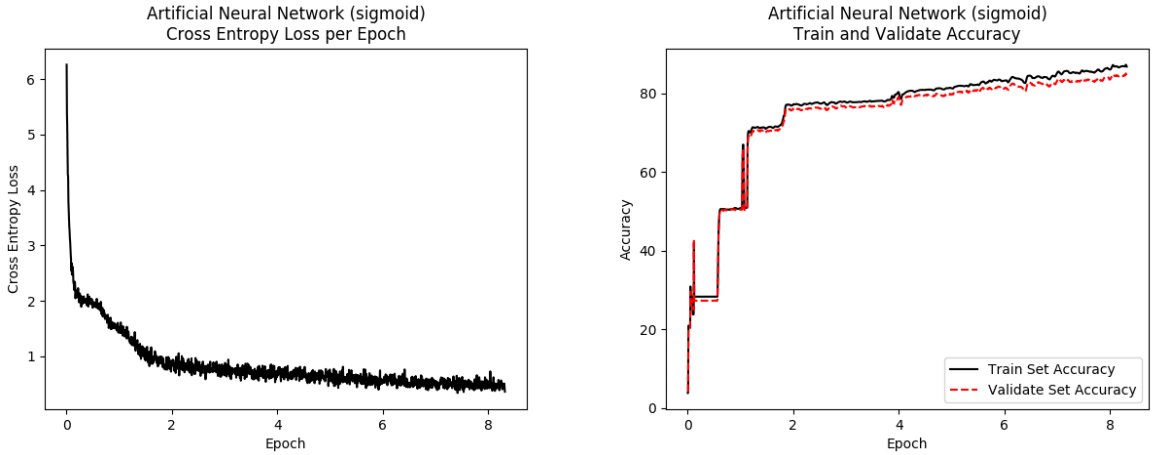
Appendix Figure 2: Loss and accuracy plots of Linear SVC



Appendix Figure 3: Loss and accuracy plots of Linear Kernel SVM



Appendix Figure 4: Loss and accuracy plots of ANN (relu)



Appendix Figure 5: Loss and accuracy plots of ANN (sigmoid)

7. Statement of contributions

Chengbo Gu is responsible for programming tasks mostly, especially for the TensorFlow applicability to this project. He has also been responsible for the SVM algorithms as well as Logistic Regression.

Zexi Han contributes to both idea generation as well as programming tasks, he proposed the initial project idea, found the resources, learned the TensorFlow and also developed the ANN models.

Hengfang Deng is mainly responsible for the writings, reviews and revisions on the proposal and the report.

All team members have actively participated the group discussion and presented strong willingness to contribute as part of a team.