Final Project
IFT6390

Myriam Laiymani (20140876)
Adel Nabli (20121744)
Pierre Rosin (20025653)
Lawrence Abdulnour (20019894)
Jean-Hughes Fournier-Lupien (Polytechnique: 1433182)

Abstract

In this final project, we tackle two of the most popular fields in machine learning: Computer Vision and Natural Language Processing. To do so, we picked 2 datasets: CIFAR-10 for image processing and IMDB movie reviews for sentiment analysis. We compare the results of 3 different models: support vector machines (SVM) as a reference model, convolutional neural networks (ConvNet) for their notoriety and ToMATo (Topological Mode Analysis Tool) for its originality.

1 Approach

One of our goals for this project was to dabble with NLP. As most of us are new to machine learning, the curiosity and willingness to apply some of the models learned to a new type of input came along. We chose to explore sentiment analysis on the IMDB movie review dataset. It consists of doing a binary classification, positive vs negative, on movie reviews. The dataset is labeled and contains 25,000 highly popular movie reviews for training, and 25,000 for testing. Text data needs further pre-processing as the algorithms we are using only deal with vector representations. We therefore needed to find a way to vectorize each text in order to benefit from a better data representation. Hence, different techniques were used to represent our data. More precisely, in our convnet and SVM models,bag-of-words representations was used beforehand and in ToMATo, a representation of the average embeddings of adjectives was opted (further details of this representation can be found in the methodology).

CIFAR-10 is probably one the most utilized dataset for machine learning research purposes. This dataset containing 32 x 32 poor resolution images permitted major and extensive scientific contributions with leading-edge accuracy results. We therefore explore an image classification problem with this dataset consisting of 10 categories, allowing us to have considerable insight on what techniques were previously done. As manifested earlier, images don't need much of a change of representation, as they are already vectorized. However, images are typically normalized, often making the convergence faster while training.

2 Methodology

2.1 Sentiment Analysis

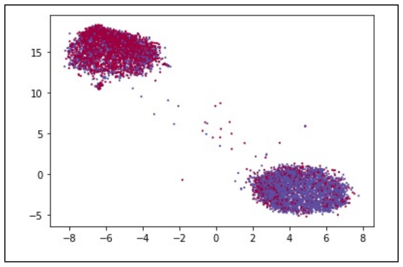
2.1.1 A ToMATo Approach to Sentimal Analysis

We were curious to test the results that would give a clustering algorithm on a classification task. But as clustering is an unsupervised task and we wanted to do a supervised one, the use of clusters to do classification isn't straight forward. The method we wanted to experiment was to detect clusters in all of our data (train and test data), and then assign to each unlabelled point the label the most seen in its cluster. But for that method to work, we have to assume that the different classes in the classification task form clear and **distinct** clusters in the vector space the input data is living in. We also have to assume that the clustering algorithm we are using will be able to detect and separate the clusters well.

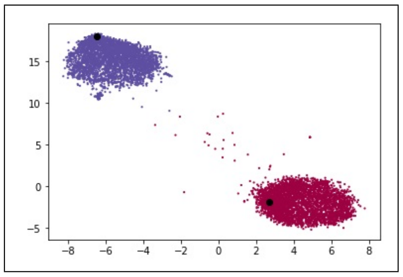
To fulfill the second assumption, we decided to implement ToMATo (*Topological Mode Analysis Tools*), a clustering algorithm that combines a graph based hill-climbing method to the idea of algebraic persistence. The reason for this choice being that the algorithm has good theoretical guarantees with regards to the number and location of the produced clusters [6].

For the first assumption, we decided to make sure the different classes produced different clusters by pre-processing our data using UMAP (*Uniform Manifold Approximation and Projection for Dimension Reduction*) [2] in a supervised way: by telling the algorithm which points we expect to see "close to each other", the dimension reduction algorithm learns a way to project the data in a smaller euclidean space that concentrate data points from the same classes in the same clusters. This algorithm is a novel manifold learning technique for dimension reduction. It is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The algorithm is a direct competitor of t-SNE [3]. According to the authors, UMAP is demonstrably faster and provides better scaling than t-SNE.

To apply this method to sentiment analysis, we finally need to represent the data (*IMDB reviews*) as vectors, in a way that will make it easy to distinguish "bad" from "good" reviews. To do so, we made the assumption that only focusing on the adjectives of each reviews would give us a good start: if the adjectives used in a review express on average a "bad" feeling, we expect the review to be a "bad" one. Thus, we extracted all the adjectives of each reviews using the `spacy` library, then we found an embedding of each adjective using pre-trained FastText embeddings (*trained on Wikipedia*), and we kept the average of those embeddings to represent each review.



(a) UMAP 2D projection



(b) ToMATo clusters

Figure 1: Projections of our test data

We tuned the hyper-parameters for both the UMAP dimension reduction (*which distance to use*) and for the ToMATo algorithm (*the merging parameter* τ) on a validation set and then splitted the dataset in only two sets (train and test). The results are given in Fig.1. Thus, we see that UMAP manage to learn an overall good projection that distinguish well the two classes. With that as input, the clustering algorithm leads to an accuracy of 77.4 %.

2.1.2 A Convnet approach to Sentimal analysis

Convnets (convolutional neural network) are designed to process data that have a grid-like topology. As images can be seen as 2D grids (or 3D if we add channels), convnets are very popular in computer vision. But, as a sequence of words can be viewed as a 1D grid, convnets are also used in NLP. With this kind of model, we obtained a 90% accuracy. It was trained exclusively on the IMBD dataset and no pretrained word embedding was used in the experiment.

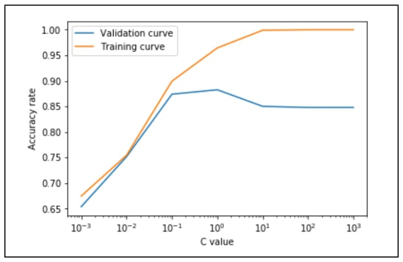
Word embedding is one the breakthroughs in NLP where each word (or phrase) is mapped to a higher dimension vector of real numbers. Similarities in meaning of words are measured by the distance between their vector representation in the new space. Mathematical operations become easier to perform in the new space.

To obtain such mapping, we chose to use the embedding layer of the high-end Keras model-level library (https://keras.io/). This layer can only be used as the first layer in a model. For a fixed vocabulary size, it provides a mapping of each word in the sequence (represented by an integer), into an output vector of a chosen length. The input sequences that are passed to the embedding layer are fixed length vector representation of the tokenized reviews: each element of the input vector is the id of a word in the vocabulary, vocabulary sorted by order of decreasing frequency of apparition in the train corpus. We fixed the sequence length to 500 based on the average length of tokenized reviews plus one standard deviation, ensuring that we have most of the information. The reviews that are shorter than 500 will be padded with 0s and the longer ones truncated. Each word is embedded into 64 length vector. The output the embedding layer of our Convnet is then 500x64. The architectural details of this network can be found in appendix 7.1)

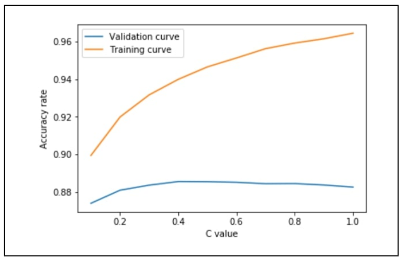
The ConvNet converges after a few epochs, going beyond 4 or 5 leads to overfitting. The accuracy of the validation sets starts dropping while the model keeps doing better on the training data. We chose early stopping as a regularization technique.

2.1.3 An SVM Approach to Sentimal Analysis

In this section we are going to experiment some SVM models in order to do a sentimental analysis. First of all we need to convert the text files into numerical feature vectors. An approach would be to use the same preprocessing as the ConvNet model but this kind of preprocessing gives us an accuracy of a Random Classifier even after 50 iterations. We are going to use two step for our preprocessing using the bag of words model. In the first step we are going to learn the vocabulary of all our documents. Then we will count the number of times a word occurs in each texts and finally assign each word to an integer counting how many time they occurred giving us a numerical feature vector for all texts. In the second step, we will give the same weightage for long documents and short documents. For this we are going to divide all values of the vector by the number of word in the document. We will also reduce the weightage of common word. All of this can be done using the scikit learn tools. We now have a good dataset ready for a sentimal analysis on a SVM. First of all we need to decide the kernel we want to use in order to have the best results. We saw that the linear kernel w as the one that best fit our dataset with an accuracy close to 85% (See Appendix 7.4). We now need to find the best penalty parameter C in order to best fit our dataset. For that we are going to use a validation set for which we will find the optimal value of C.



(a) value of C between 0.001 and 1000



(b) value of C between 0.1 and 1

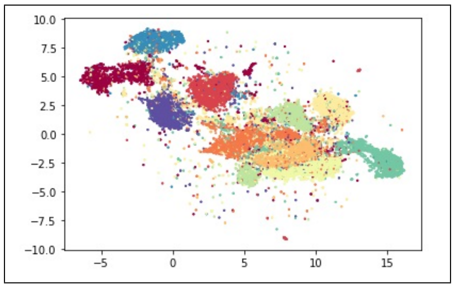
Figure 2: Plots of accuracy of different value of margins

The optimal value of C is 0.4. Before this value the model is under fitted and after this value it is over fitted. In conclusion we found that for a linear kernel with a C equals to 0.4 we can reach an accuracy of 88.53% on a sentimal analysis.

2.2 CIFAR-10

2.2.1 A ToMATo Approach to CIFAR-10

Contrary to the sentiment analysis dataset, the CIFAR-10 directly provides a vectorized representation of the data. So the only steps remaining for us are the supervised UMAP projection of the dataset and the creation of the different clusters using ToMATo.


(a) UMAP 2-D projection of the training data

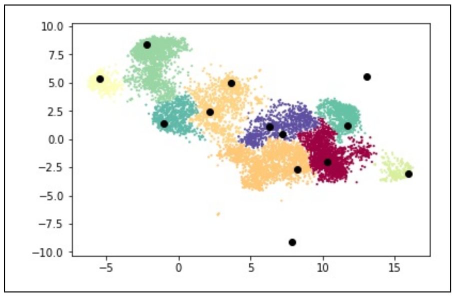

(b) ToMATo clusters on the test data

Figure 3: UMAP 2-D projections on CIFAR-10

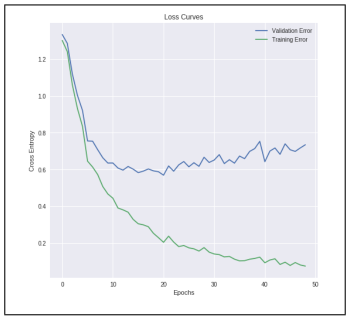
The problem that directly emerge from those figures is that UMAP doesn't manage to clearly form separated clusters on CIFAR-10 (*several regions corresponding to different classes are gathered in the same big cluster*). As one of our assumptions for using a clustering algorithm was that the different classes formed easily distinguishable clusters doesn't hold on this dataset, we didn't go further with this approach here (the clusters found being meaningless). But, as UMAP still manages to form different regions of the space corresponding to different classes, we can expect that learning the decisions boundaries using an SVM would produce good results.

2.2.2 A Convnet Approach to CIFAR-10

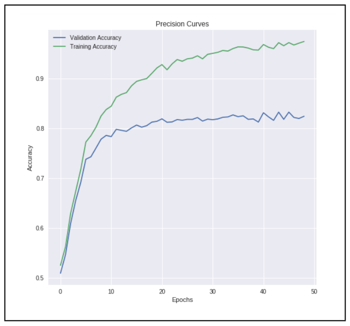
A 80/20 train/valid separation scheme was used, as it is a common distribution for datasets of 50,000 training images. Prior to fitting a convolutional neural network, data augmentation is performed on the dataset using the torchvision transform module. As contrary to generating new examples, torchvision.transforms applies a new set of random transformations to the dataset in any epoch. In other words, it is showing a variant of our dataset at every iteration, instead of feeding the same enlarged dataset a number of times. Basic transformations were thus chosen such as horizontal flips and random rotation from a range of [-20, 20] degrees. It is also significant to mention that only one pre-processing step was administered to our dataset. Indeed, normalization of image inputs was done to ensure that each pixel input has a similar data distribution. The preceding is done by subtracting the mean from each pixel and dividing it by its standard deviation, and this for all three RGB data inputs. ToTensor's pytorch function then scales the inputs in the range of [0,1].

To address the CIFAR-10 image classification problem, a dense convolutional neural network, inspired by the state-of-the-art architecture adopted by Simonyan and Zisserman [1], was introduced. Inputs pass through a total of 8 convolutional layers, 5 max-pooling layers and a fully connected layer. Kernel sizes of convolutional layers are 3 x 3, accompanied with a stride of 1, thus leading outputs to have the same witdh size as the inputs. Max-pooling is also executed over a 2 x 2 window with stride 2. Moreover, each convolutional layer is followed by a normalization layer as well as a rectification ReLU non-linearity (further details of this architecture can be found in appendix 7.3).

Although a relatively big number of epochs would have seemed appropriate for such a dense network, our model was trained throughout 50 epochs. As noticeable in figure 4, we can see that the accuracy curve of the valid set, is slowly reaching a plateau and hence, training over more epochs would have not been of use. Finally, a learning rate and a batch size of respectively 0.005 and 64 were utilized during the training process along with a common Adam optimizer, resulting in a 83% accuracy performance on the test set.



(a) Loss curves

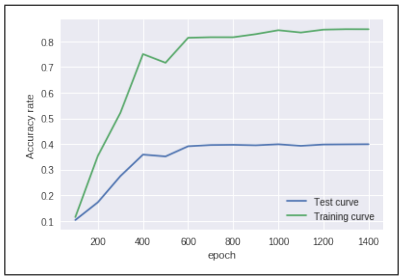


(b) Accuracy curves

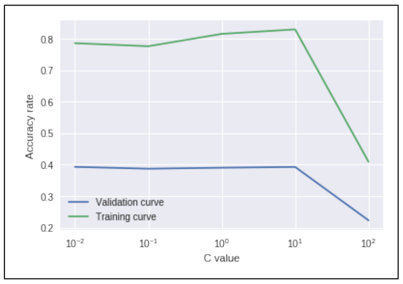
Figure 4: Dense Neural Network

2.2.3 An SVM Approach to CIFAR-10

For reference, we compare the previous approach with the standard SVM algorithm. The dataset is firstly reshaped to vector before being normalized using the *StandardScaler()* function from the *sklearn* library. Prior to the classification, the dimension of the data input is reduced to 2 dimensions using the UMAP algorithm [2]. We then run the model for different numbers of iteration and penalty hyper-parameter C on the validation set, to better choose the optimal values. Furthermore, the rbf (radial basis function) kernel was chosen, since this hyper-parameter led to best results. Figure 5a shows the evolution of the accuracy on the train (train curve) and validation set (test curve). The accuracy quickly reaches a plateau around 600 iterations. In order to avoid overfitting, we fixed the number of iterations to 800. Figure 5b presents the optimization of the hyper-parameter C. We fix C to 1. The best model reached 40.55% on the test set. Due to the dimensionality reduction that we conducted, the data can be visualized in a 2-D graph. These graphs are presented in figure 9a and 9b.



(a) Optimization of the number of iterations with $C = 1$.



(b) Optimization of the penalty parameter C with 800 iterations.

Figure 5: Dense Neural Network

3 Comparison and Recommendations

As mentioned previously, the dense network for the CIFAR-10 dataset achieved a 83% accuracy on a test set containing 10,000 images. In order to see which categories undermined the overall performance of our model, a confusion matrix was produced, thus permitting us to compare accuracies of different classes (see Figure 6). Ironically enough, the class that actually dragged down the most our overall precision were cats, whereas cats are probably one the most studied images in classification. This last statement can be easily explained by the fact that dogs and cats are similar in appearance and our model has mistaken a lot of the cats for dogs (true label cats and predicted label dogs = 202). A similar reasoning can be made for car and trucks; a lot of the cars were mistaken for trucks and vice versa.

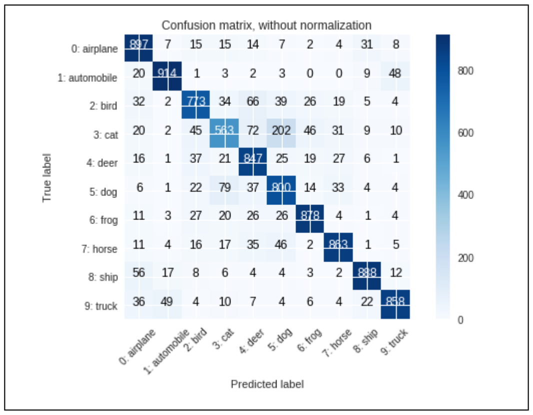


Figure 6: Confusion Matrix

It is by no surprise that in both datasets, convolutional neural networks provide superior results to other methods, as they are deep and complex architectures. However, we seem to notice that ConvNet's gave much more leverage for the CIFAR-10 dataset, confirming once again that convolutional neural networks are well-suited for grid-like data inputs, such as images. Indeed, the ConvNet for the sentiment analysis resulted in an efficient 90% accuracy, however the SVM also lead to honorable events (88.53% accuracy). As contrary, we find a much greater gap between the two with CIFAR-10 (83% accuracy for ConvNet and 41% accuracy for SVM), hence reiterating the importance of using more refined solutions with images (such as ConvNets). Another explanation for such a difference would be that the sentiment analysis dataset carried a binary classification problem, whereas CIFAR-10 requires to classify our inputs into 10 categories. This essentially translate into the sentiment analysis having more examples per class. Furthermore, because the CIFAR-10 has more classes, feature extraction becomes critical to point out the subtle distinctions between these classes and ConvNets actually learn feature maps throughout the training process. In conclusion, considering the outcomes illustrated in table 1, an obvious choice of convnet's for both datasets seems best.

Dataset	ToMATo	SVM	ConvNet
IMDB	77%	89%	90%
CIFAR-10	-	41%	83%

Table 1: Comparison of the accuracy of the different algorithms on the test set

Further work/developments for the CIFAR-10 dataset can be achieved by performing 'Network engineering', i.e. trying variants of convolutional architectures, such as residual neural network and recurrent neural network. One state-of-the-art achievement leading to a 95.16% accuracy, explained in the paper Dual Path Network [4], actually benefited from a dual path architecture, where densely and residual paths jointly work together. Another well-acclaimed method was made public by Ben Graham [5], also getting comparable results (95.53% accuracy on the CIFAR-10 kaggle competition), where the utilization of fractional max-pooling was adopted. Moreover, for sentiment analysis, recurrent neural networks could be a great alternative to our current model, as they create an understanding of the past and the present in inputs at the same time. This is in fact a critical property for natural language processing problems, since the whole context of the input is important when trying to understand its semantic.

4 Acknowledgements

Adel implemented the ToMATo algorithm, testing it in on both datasets and compared the post-UMAP pre-processing results with an SVM. Myriam and Pierre worked on the NLP part of the project, comparing the ConvNet and SVM for the purpose of classifying the movie reviews. Lawrence and Jean-Hughes worked on image classification problem of the Cifar-10 dataset and tested several architectures of ConvNet's.

5 References

[1] Simonyan K., Zisserman A. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015 Conference Paper. 1-14

[2] McInnes, Leland, and John Healy. "Umap: Uniform manifold approximation and projection for dimension reduction." arXiv preprint arXiv:1802.03426 (2018).

[3] Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." Journal of machine learning research 9.Nov (2008): 2579-2605.

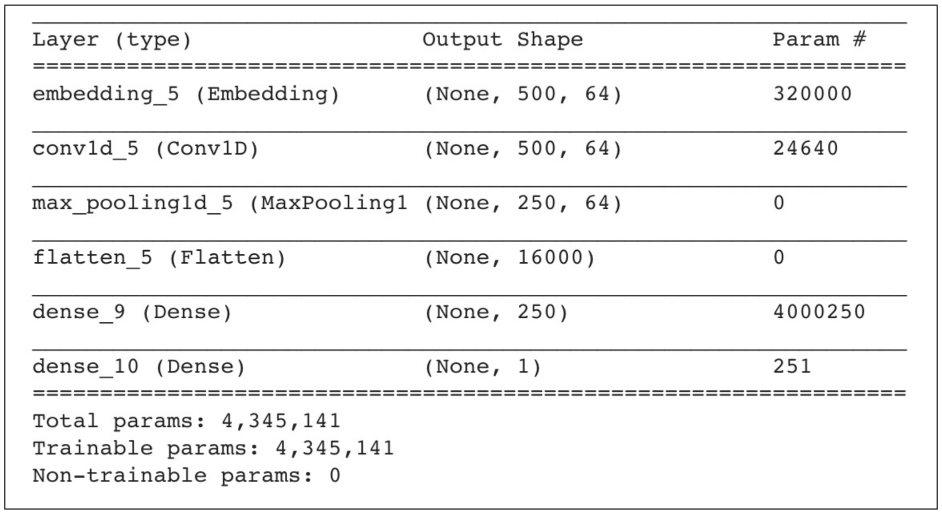
[4] Chen Y., Li J., Xiao H., Jin X., Yan S., Feng, J. Dual Path Networks. *arXiv preprint:arXiv:1707.01629*, 2017

[5] Graham B. Fractional Max-Pooling *arXiv preprint:arXiv:1412.6071*, 2017

[6] Steve Oudot. "Persistence Theory: From Quiver Representations to Data Analysis". AMS Mathematical Surveys and Monographs, volume 209, 2015.

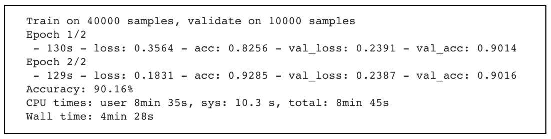
6 Appendix

6.1 Architecture of the CNN used for IMBD



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_5 (Embedding)      (None, 500, 64)           320000    
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 500, 64)           24640     
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 250, 64)           0         
_________________________________________________________________
flatten_5 (Flatten)          (None, 16000)             0         
_________________________________________________________________
dense_9 (Dense)              (None, 250)               4000250   
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 251       
=================================================================
Total params: 4,345,141
Trainable params: 4,345,141
Non-trainable params: 0
```

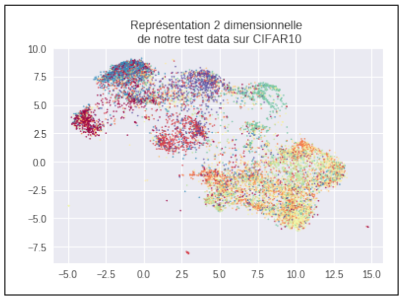
Figure 7: CNN Model Layers for Sentiment Analysis



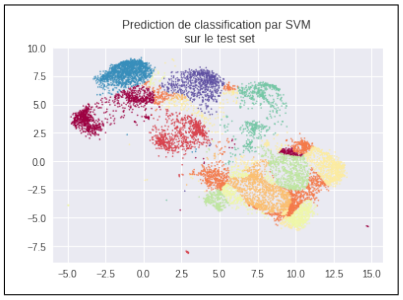
```
Train on 40000 samples, validate on 10000 samples
Epoch 1/2
 - 130s - loss: 0.3564 - acc: 0.8256 - val_loss: 0.2391 - val_acc: 0.9014
Epoch 2/2
 - 129s - loss: 0.1831 - acc: 0.9285 - val_loss: 0.2387 - val_acc: 0.9016
Accuracy: 90.16%
CPU times: user 8min 35s, sys: 10.3 s, total: 8min 45s
Wall time: 4min 28s
```

Figure 8: Convergence of the CNN

6.2 Visualization CIFAR10 test set and the prediction made by the SVM algorithm in 2-dimensions.



(a) 2-D represention of the test set after UMAP algorithm, colorcoded with the labels.



(b) 2-D represention of the test set after UMAP algorithm, colorcoded with the predictions of the SVM algorithm.

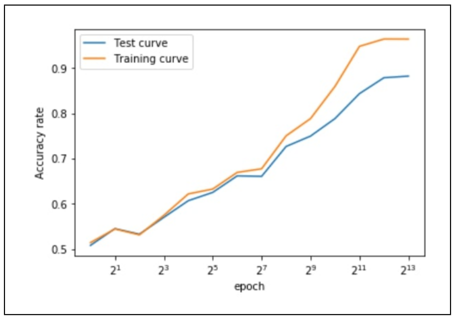
Figure 9: Visualization of the data and the prediction in 2 dimension.

6.3 Details of the Dense Network Architecture for CIFAR-10

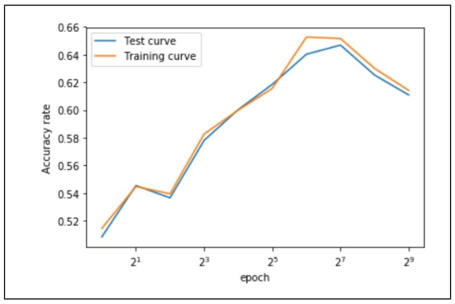
Layer	Kernel	Filters	Stride	Padding	Output Size
ConV/ReLU MaxPool	3x3 2x2	64	1	1	64x16x16
ConV/ReLU MaxPool	3x3 2x2	128	1	1	128x8x8
ConV/ReLU MaxPool	3x3 -	256	1	1	256x8x8
ConV/ReLU MaxPool	3x3 2x2	256	1	1	256x4x4
ConV/ReLU MaxPool	3x3 -	512	1	1	512x4x4
ConV/ReLU MaxPool	3x3 2x2	512	1	1	512x2x2
ConV/ReLU MaxPool	3x3 -	512	1	1	512x2x2
ConV/ReLU MaxPool	3x3 2x2	512	1	1	512x1x1
Fully Connected Layer	-	-	-	-	512x10*

Table 2: ConvNet Architecture for CIFAR-10

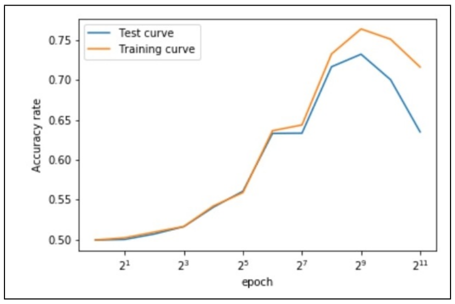
6.4 Testing kernels for the sentimal analysis SVM



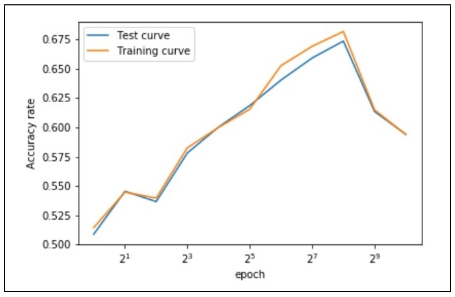
(a) linear kernel accuracy



(b) sigmoid kernel accuracy



(c) linear kernel accuracy



(d) Radial basis function kernel accuracy

Figure 10: plots of accuracy of different kernels for the sentimental analysis SVM