THE UNIVERSITY OF
SYDNEY

School of Information Technologies

The University of Sydney

COMP5318 – Assignment 2

Classification Model

CIFAR10

By

Cameron Wasilewsky
Kristopher Lopez
Arjun Sathasivam

Submitted: 5th of June 2017

Student IDs:

cwas5136 – 470267240

klop5007 – 470236420

asat9100 – 470367366

Originality Statement

We hereby declare that this submission is our own work and to the best of our knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at The University of Sydney or any other educational institution, except where due acknowledgement is made in the document. Any contribution made to the research by others, with whom we have worked with, is explicitly acknowledged in the document. We also declare that the intellectual content of this document is the product of our own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed – Cameron Wasilewsky - …….....

Signed – Kristopher Lopez - …….....

Signed – Arjun Sathasivam - …….....

Date Monday 5th of June 2017 …….....

Contents

List of Figures

List of Tables

Chapter 1: Abstract

Artificial Intelligence (AI) presents itself as the technology that will alter the quality of living for all humans. Like the steam engine or the internet, AI will revolutionise every aspect of humanity. However, that point is still in the future, at present AI research has been motivated by computer vision problems and its application to areas ranging from digit recognition to autonomous driving. Convolutional neural networks (CNN) have dominated research for object recognition problems since their introduction in 2012. CIFAR-10[1] is no exception and has seen CNN variants with results ranging up to 96.53% [1]. This paper explores and compares popular pre-CNN methods and vanilla CNN implementation to assess the significance of pre-CNN methods. A vanilla CNN model was found to outperform all models by 200%, achieving an accuracy of 82%. This result was a depiction of the true power of deep learning (DL) technologies.

[1] Image classification of 10 common objects

Chapter 2: Introduction

1 Problem

CIFAR-10 is a well-known collection of low resolution, labelled images consisting of ten common objects. This body of research attempts to maximise the accuracy of identifying each image's correct class. Thus, the problem is one of computer vision, how well can software identity an object. This task revolves around complexities and difficulties with large data sets, complex matrix structures and analytics models that can handle a high number of parameters.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class [2].

Figure 1: Example images included in the CIFAR-10 dataset

These low-resolution images present a challenge for researchers to label each of the test images correctly despite the relatively low number of features. Attempts to manually classify the dataset led to 96.4% accuracy with the images to the right in Figure 1, incorrectly labelled. Common human errors include misclassifying horses and deer, automobiles and trucks.

2 Importance

To allow computers, software and AI to interact with humans and be beneficial it is essential for them to able to view, hear and communicate with the world. This research focuses on a machine's ability to interpret, understand and identify visual elements within its environment. Although this data set is simplistic the techniques and models can be applied to more complex and valuable problems. Techniques used in this report are currently in practice within facial recognition software, self-driving cars and security systems. Thus, understanding this complex concept and the best way to approach and solve the problem is essential in progressing with processing camera footage and photo repositories.

The low resolution and close similarity between several pairs of classes creates a need for the researcher to discover techniques that can draw a distinction between these classes with a relatively low number of features. Challenges in computer vision and other machine learning problems include:

- creating a model that can correctly classify given sparse data or a low number of features
- create a model that can learn from limited samples

In this paper, the use of popular pre-CNN models as well as a CNN will be used to determine whether or they are still relevant. The pre-CNN models that will be explored are:

- Logistic Regression (LR)
- Support Vector Machine (SVM)
- Linear Models w/ Stochastic Gradient Descent (SGD)
- K Nearest Neighbours (kNN)
- Random Forest (RF)
- Gradient Boosting Machine (GBM)

Chapter 3: Previous Work

1 Introduction

CIFAR-10 is a well-known curated data set. Due to its ubiquity, ease of use and simple structure it has been used in a range of papers focusing on computer vision. Chapter 3 provides details on some of the more recent research.

2 CNN Implementations

There has been a great deal of research performed on the CIFAR-10 dataset. The below papers were of interest to this study:

- CIFAR-10: KNN Based Ensemble of Classifiers [3].
- Regularization of Neural Networks using DropConnect [4].
- Multi-column Deep Neural Networks for Image Classification [5].
- Recurrent Convolutional Neural Network for Object Recognition [6].

Abouelnaga, and Hager's [3] paper explores many different CNN architectures with results ranging from 93.33% to 93.99% using features such as max pooling and drop out across 8 convolutional layers and 3 linear layers (Figure 2). The paper compares the results of pre-CNN models with CNN publishing results from 37.5% (Logistic Regression) to 49.52% (Random Forest) and demonstrates the significance of the

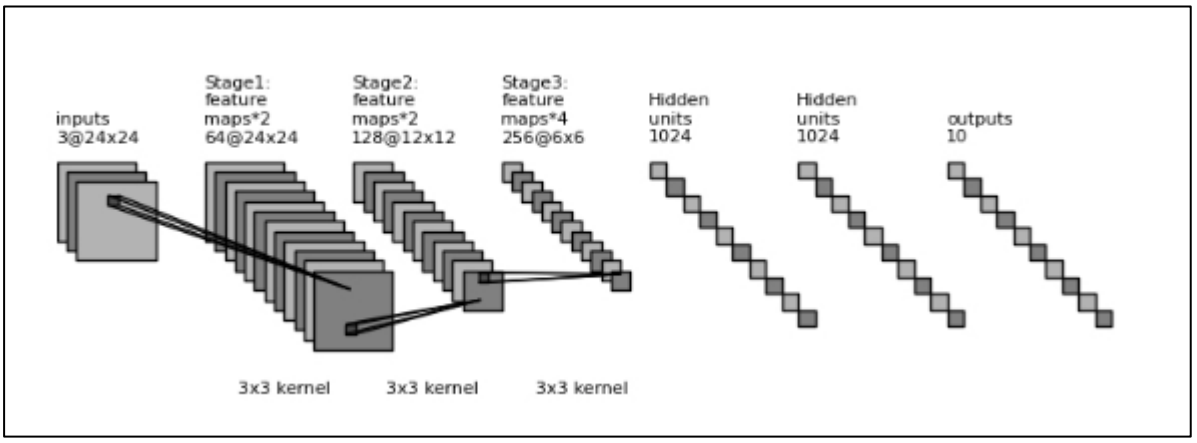


Figure 2: CNN Model used to achieve above 93% accuracy

ensemble showing an increase to 94.02% when combined with KNN.

Despite the relatively low improvement, from 93.99% to 94.02%, the result is still significant in a world where state-of-the-art improvements diminish as models move past human-level accuracy and closer toward 100%.

As shown in the paper 'Regularization of Neural Networks using DropConnect' state-of-the-art results in 2013 on the MNIST dataset [5] rose to 99.79% compared with a ~99.80% human performance. DropConnect, a variation of Geoff Hinton's [4] DropOut, reaches a 91.64% accuracy on CIFAR-10 in comparison to 91.17% when DropOut is used (Figure 3). The multi-columnar approach [1], a deep convolutional neural network, only manages 89.79% accuracy, demonstrating the tweaks and improvements made to CNN architectures since their popularisation in 2012.

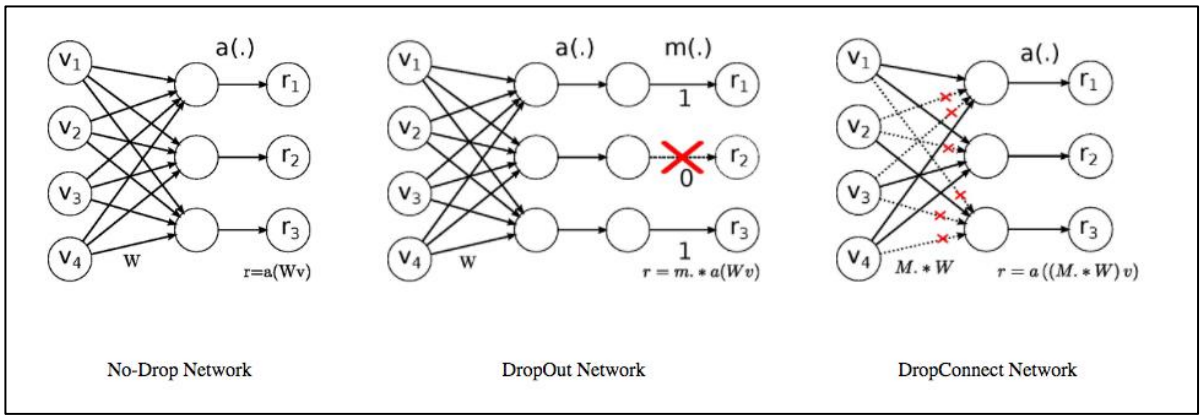


Figure 3: Depiction of drop out network architecture

The paper 'Recurrent Convolutional Neural Network for Object Recognition' [6] introduces the concept of a recurrent CNN (RCNN) for the purposes of object recognition, which combines the traditional feed-forward CNN with a recurrent multilayer perceptron. This model was proposed to better handle the contextual features

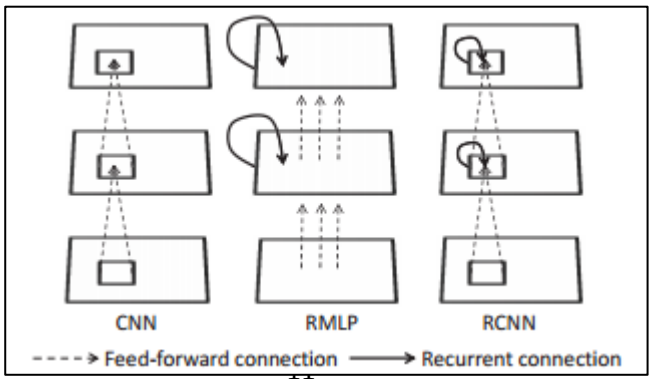


Figure 4: Recurrent CNN architecture

that humans typically rely on when identifying objects.

The model, when run on the CIFAR-10 data set, produced an accuracy of 92.91%, which performed comparably to other state-of-the-art results in 2015.The result made use of dropout and data augmentation (using translations and horizontal reflections) to achieve the outcome reported. The RCNN approach also performed well on the MNIST data set, achieving an accuracy of 99.69%. While not outperforming other methods available, it is an impressive result regardless.

Chapter 4: Methods

1 Introduction

Chapter 4 analyses and outlines the three model types that were explored in this research task. Section 2 and 3 use models contained within the Scikit Learn library. Section 2 focuses on linear models; Logistic Regression (LR), Linear SVC (Linear Kernel SVM) and SGD Classifiers (linear models optimised with stochastic gradient descent). The third section discusses K-Nearest Neighbours, Random Forest and Gradient Boosting Machines, three non-linear models. The final model discussed is Convolutional Neural Network (CNN) a deep learning architecture. Further detail, visualisation and mathematical analysis of each model can be seen in Appendix 3, 4 and 5.

2 Linear Models

2.1 Theory

The linear models explored in this paper are binary classification models which use 'one-vs-all' to extend to the multi-class case. LR, which can be estimated by SVM and Neural Networks, uses a sigmoid function in its attempt to separate each class. SVM looks to maximise the margin of a separating hyperplane projected into infinite dimensional space with a linear kernel.

2.2 Pre-processing

Linear models often underperform when trained on a dataset containing complex non-linear relationships. Pre-processing, such as Principal Components Analysis (PCA) is often required to reduce this complexity, discovering directions in the data that represent the most variance in the data. An extensive grid search was used on varied levels of principal components to discover separability from the full 3072 dimensions. As can be seen in Figure 5, the first 10 principal components capture 65% of the variance found within the dataset, 100 components captures 90% and 500 components captures 98%.

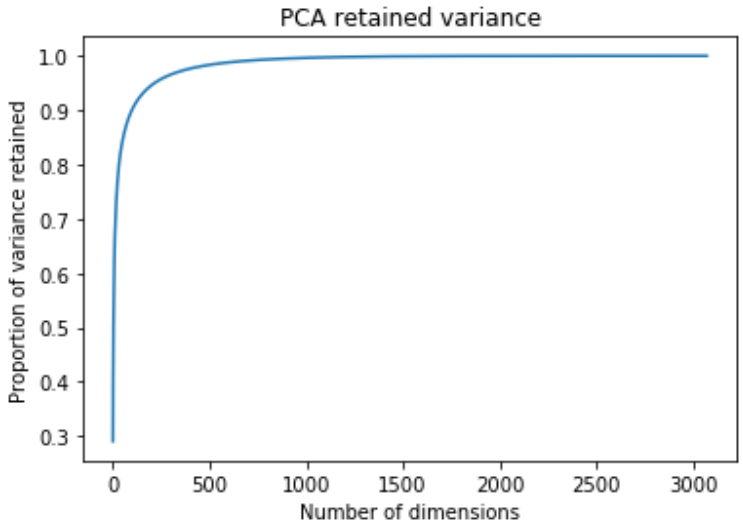


Figure 5: PCA variance by dimensions

2.3 Design choices

In addition to the grid search across principal components, using 10-fold cross validation, several hyper-parameters were explored to discover the best performing linear models on the CIFAR-10 dataset. These include search on models with and without l2 regularisation, regularisation parameter 'c', different loss functions and convergence using stochastic gradient descent.

The results of the grid search for each of the linear model options are displayed below as a heat map, with both LR and SVM (with squared hinge loss) revealing insight. Across both models it is clear that for optimal performance at least 200 principal components retaining ~95% variance are required, which also reveals that almost all of the data's information is useful in separation. Meanwhile, there is a loss in performance when using more than 2000 components, the final 1072 components representing 0.02% variance.

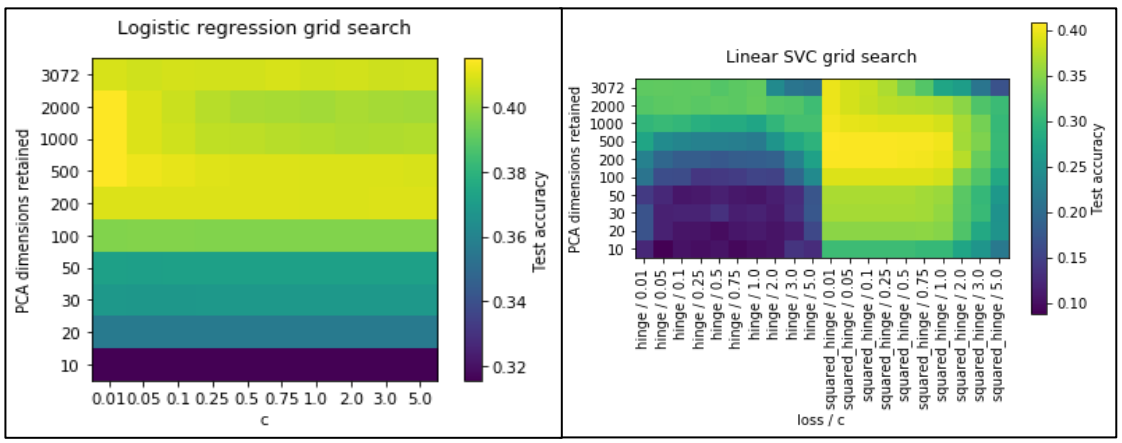


Figure 6: LR & SVC Hyperparameter Heat map, where darker blue is lower accuracy

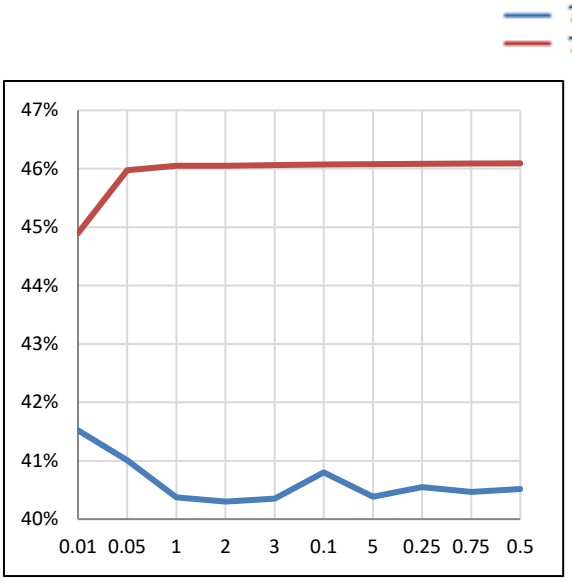


Figure 7: LR grid search over 1000 principal components

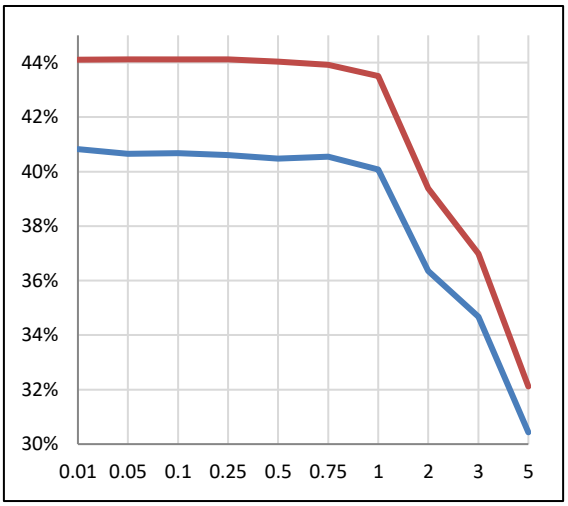


Figure 8: SVM grid search over 500 principal components with squared hinge loss

Sklearn's SGD Classifier was also used to iterate over a large number of parameters. This model optimises linear models with stochastic gradient descent but failed to find a combination of hyper-parameters that out-perform LR and SVM.

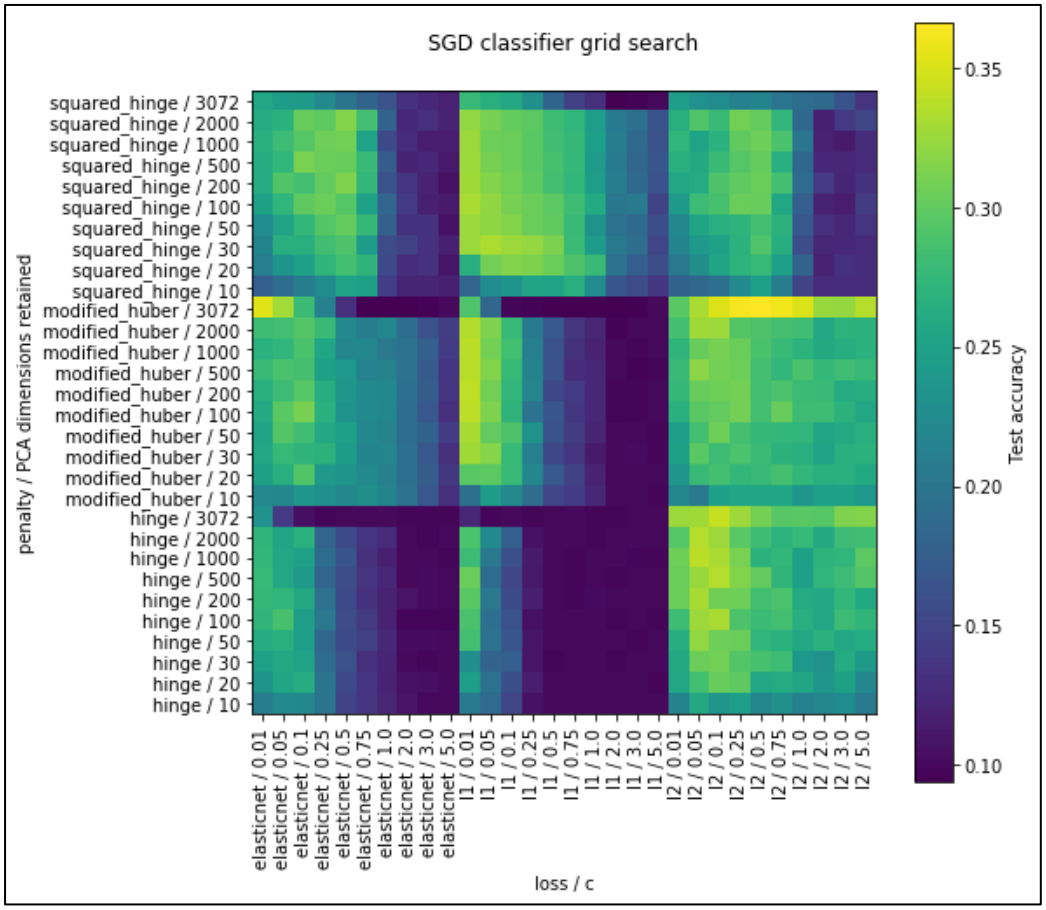


Figure 9: SGD Hyperparameter Heat map, where darker blue is lower accuracy

3 Non-Linear Models

3.1 Theory

Nearest Neighbours and Random Forest both create non-linear decision boundaries. Following the conversion of each image into a vector with normalised entries, nearest neighbours evaluates the distance of each new image to the set of training images. A vote is taken from the class label from the 'k' nearest neighbours and the majority vote used for prediction. Likewise, the Random Forest also takes a majority of vote, but does so by creating an ensemble of decision trees created on different representations of the data.

3.2 Pre-processing

As with the linear models, PCA was used to reduce the size of the feature set. Given the relative computational load of running a random forest model, PCA was necessary to allow a reasonable run time, particularly for models where 1000+ estimators were trained in the ensemble.

3.3 Design choices

For Nearest Neighbours (KNN), a grid search was used over principal components and k-neighbours (Figure 11). The heat map reveals that the algorithm works best when trained on the first 20 principal components (~75% variance) and taking a poll from the first 20 – 30 nearest neighbours. Interestingly, KNN requires both the low dimensionality and over-fitting qualities of low neighbours to produce results comparable to the linear models.

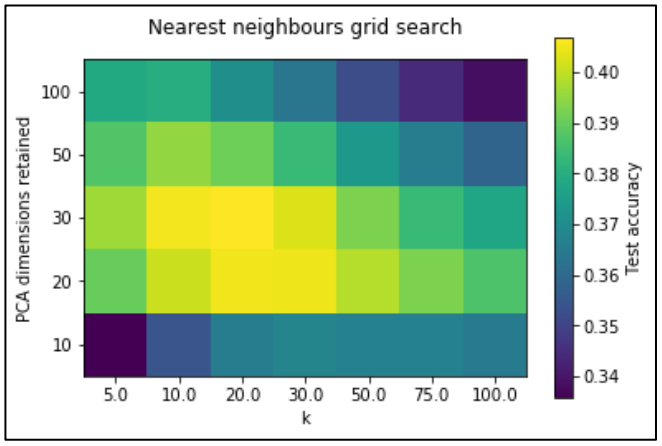


Figure 11: KNN Hyperparameter Heat map

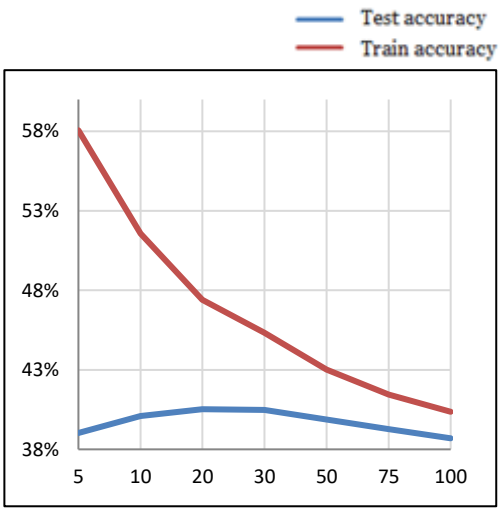


Figure 10: KNN Hyperparameter Grid search over 20 principal components

Of the Scikit-learn implementations [7], Random Forest model was only second to the Gradient Boosting Machine (GBM) with the most successful with results pushing up to 45% compared to 40% for SVM, 42% for LR and KNN. Grid search was performed over the number of estimators to include in the ensemble (number of trees to include in the forest), the percentage of features to use for each tree and the maximum depth for each tree. The search revealed that for this increase in performance, the ensemble requires trees with at least a depth of 10, indicating that the dataset is complex and requires an array of weak, yet complex learners to improve performance. Whilst not as impactful, when each tree used between 5% and 75% of the features chosen at random, performance was optimised.

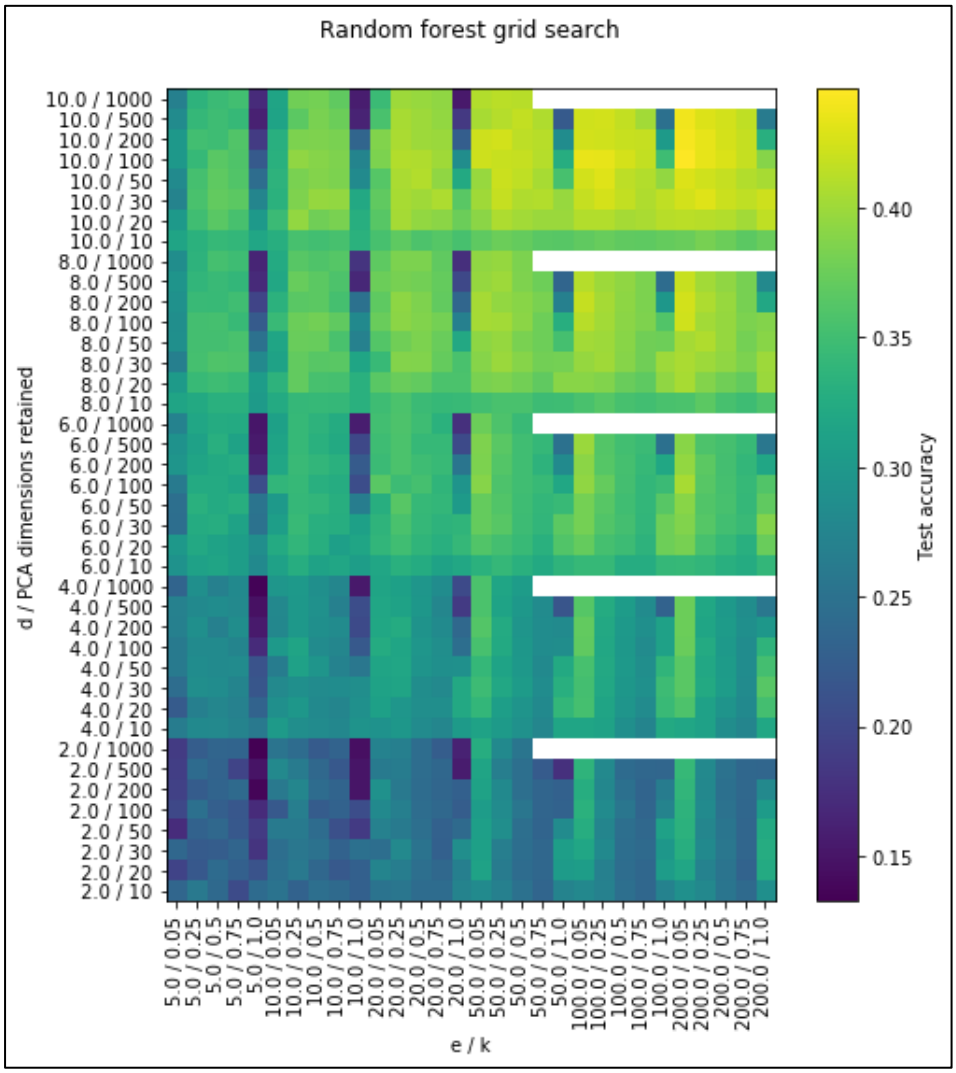


Figure 12: Random Forest Hyperparameter Heatmap, where darker blue is lower accuracy

The final model explored within Scikit-learn was the Gradient Boosting Machine (GBM). Like RF, GBM is an ensemble classifier that builds a series of weak learners which then poll a vote to determine which class to assign. Whereas RF builds a decision tree on random samples of the data, GBM iteratively builds weak learners on the residuals left behind by earlier learners. An extensive grid search was not possible for GBM, with models taking close to 50 minutes per iteration (as opposed to 3 minutes for RF), using the tuned hyperparameters from RF indicated a significantly lift to 50%+ in test accuracy.

4 Convolutional Neural Network (CNN)

4.1 Theory

Unlike the aforementioned techniques, A CNN is a deep learning model. It utilises interconnected nodes of image filers to extract and interpret key features within an image allowing for object identification. CNNs are computationally more intensive, requiring a large number of matrix calculations to achieve a result. The advent of GPU paralysation has allowed for this time to be cut down to a fraction. As a result, CNNs have become more main stream and are being developed rapidly.

4.2 Pre-processing

The model benefits from a broader range of dimensions, as a result pre-processing for CNN's typically involves data augmentation (increasing the data set) rather than feature reduction. In this experiment, a simple application of image rotations and blur was applied, this was not successful resulting in half the accuracy (Table 1). However, normalisation of the pixels to grey scale was applied, to reduce multiplication sizes.

Model #	Process.	Dimensions	Time to Train (s)	Time to Test (s)	Accuracy	F1 Score	# of Convolution Layers	Filter Sizes	Epochs
1	GPU	All	4257	19	0.821	0.820	6	48,96,192	200
2	GPU	All	4145	13	0.803	0.800	3	32,64,128	200
3	GPU	All	10334	23	0.799	0.800	6	96,192,384	200
4	GPU	All	567	24	0.819	0.820	6	48,96,192	25
5	GPU	All	1106	29	0.811	0.810	6	48,96,192	50
6	GPU	All with Rotation and Blur	4684	19	0.433	0.428	6	48,96,192	200

Table 1: CNN Hyperparameter and model structure comparison

4.3 Design choices

The complexity and variation in CNNs are vast, thus development of the model was not a simple grid search approach. Two base model architectures were developed, tested and then the higher performing was tuned slightly. The initial models were similar, the first had 3 convolutional layers, and the second 6, both finishing with a fully connected classifier. Initial testing indicated higher performance with the more complex model, and it was thus tuned. The design choices for such a model focus around the number of filters per block, epochs or training and the training rate. Table 1 outlines the results of the models that were tested. Figure 13 depicts the accuracy of Model One during the training process. Although it was the highest performing it was clear that it was overfitting, and reached a plateau of improvement. Therefore, Models Four and Five were tested with reduced training epochs to improve timing.

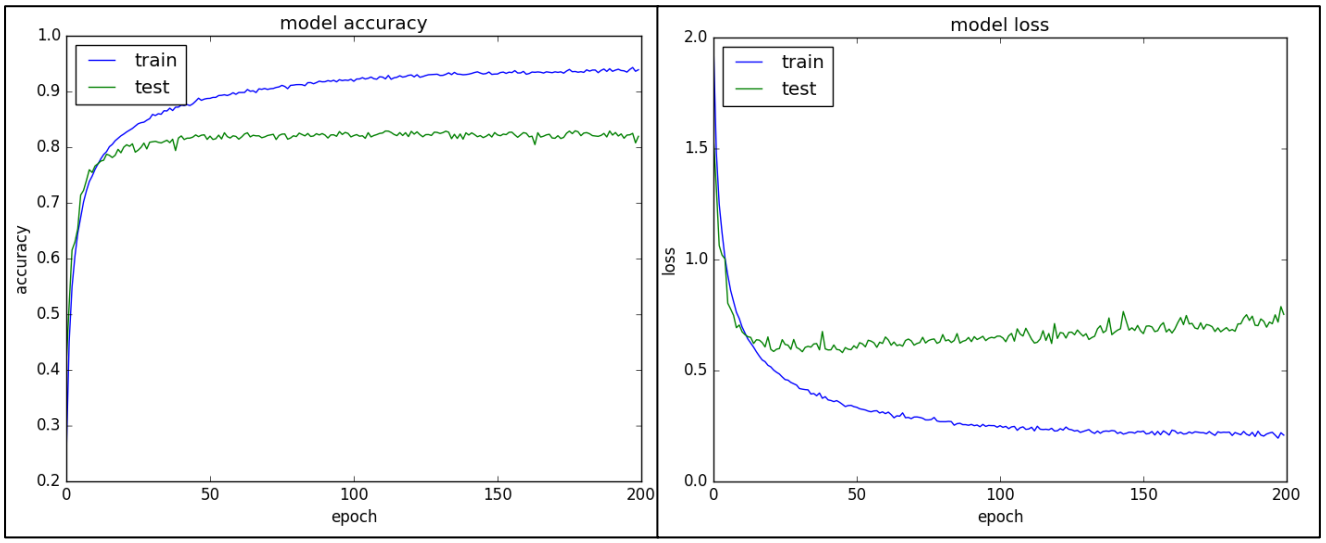


Figure 13: CNN Model 1's training history [11]

Model Four was chosen as the best performing model due to its quick time to train on a GPU instance and its high accuracy. Its architecture can be seen in Figure 14. Finally, to ensure the model was working as expected internally the six convolutional layers were visualised the results of which can be seen in appendix 5. The visualisations indicate the increase in complexity of layers as they begin to focus done onto features.

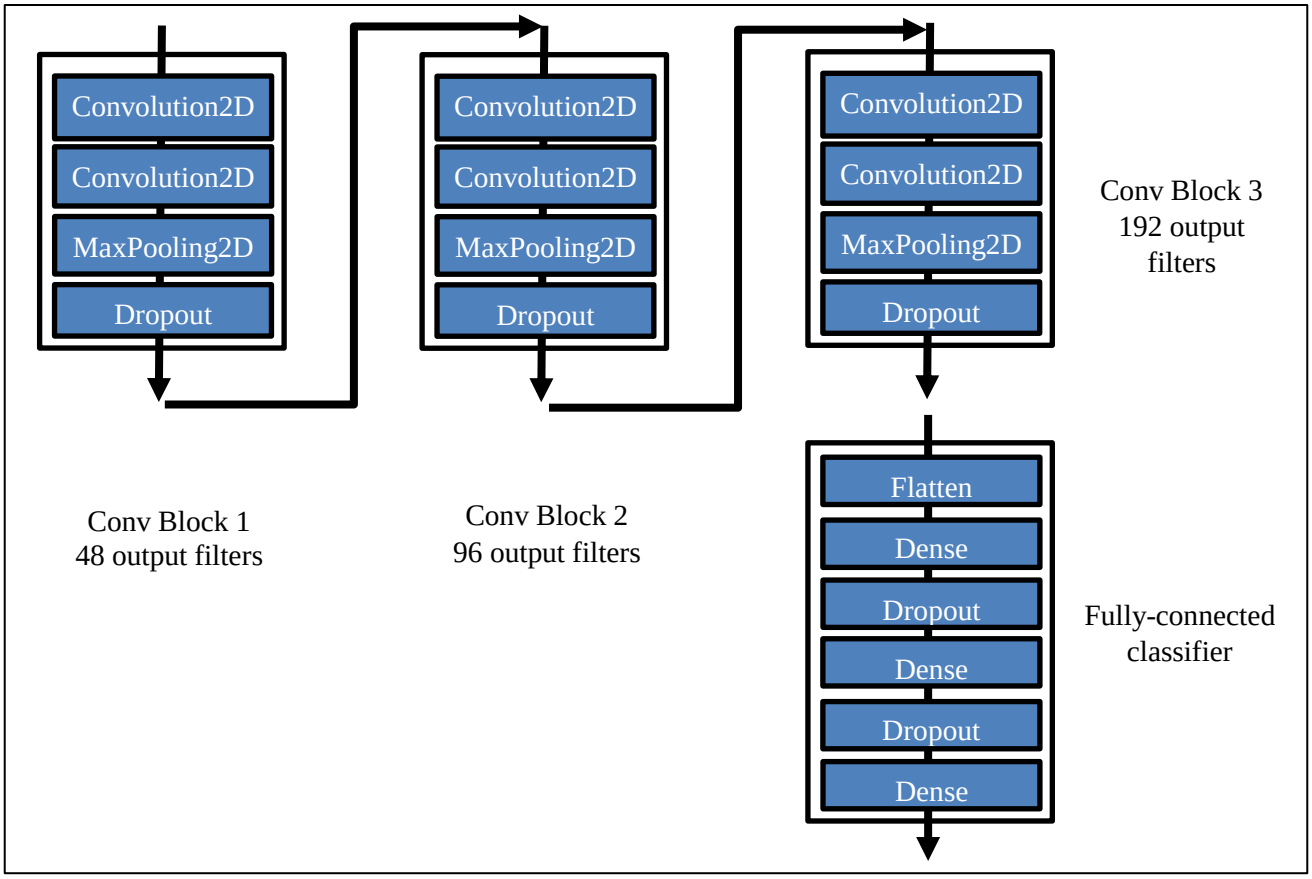


Figure 14: CNN Architecture, a description of each element can be found in Appendix 4

Chapter 5: Experiments and Discussion

1 Introduction

From the models discussed in chapter 4 the best performing of each section were chosen for deeper analysis, understanding their performance by examining summary tables and the associated confusion matrix. The chapter concludes by discussing and comparing the models' performance, processing requirements and complexity.

2 Linear Models

2.1 Logistic Regression (LR)

After performing an extensive grid search over combinations of regularisation type, regularisation parameter and principal components, using 10-fold cross validation the below hyper-parameters were discovered as the highest performing:

- 1000 principal components
- l2 regularisation
- Regularisation parameter 'c': 0.01

When training on the 50000 provided samples with the above hyper-parameters the confusion matrix and associated metrics produced (precision, recall and f-score) are given in Figure 15 and Figure 16.

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	498	45	45	34	24	26	24	52	176	76
	automobile	58	492	24	35	23	38	39	50	80	161
	bird	98	44	287	91	119	75	153	66	44	23
	cat	41	61	100	265	53	192	134	52	36	66
	deer	58	24	138	73	299	85	164	105	23	31
	dog	45	41	92	159	76	355	88	72	47	25
	frog	16	42	67	119	97	80	500	36	16	27
	horse	49	50	62	63	86	80	41	461	32	76
	ship	151	74	23	23	10	48	9	20	543	99
	truck	67	183	21	27	16	25	47	50	90	474

Figure 15: LR Confusion Matrix

	precision	recall	f1-score
airplane	0.4607	0.4980	0.4793
automobile	0.4659	0.4920	0.4790
bird	0.3341	0.2870	0.3106
cat	0.2981	0.2650	0.2815
deer	0.3724	0.2990	0.3357
dog	0.3536	0.3550	0.3543
frog	0.4170	0.5000	0.4585
horse	0.4782	0.4610	0.4696
ship	0.4995	0.5430	0.5213
truck	0.4480	0.4740	0.4610
avg/total	0.4128	0.4174	0.4151

Figure 16: LR Results Summary

The logistic regression model falls well short of the human benchmark (96.4%), with almost all classes having less than 50% precision and recall and an overall accuracy of 41.74%. LR manages to classify non-sentient objects relatively well, while heavily misclassifying sentient objects (aside from horses). Even with this linear model the confusion matrix reveals the closeness between certain classes, with airplanes often misclassified as ships and automobiles as trucks.

2.2 Support Vector Machine - Linear Kernel (SVM)

Grid search found the below hyper-parameters for the SVM model (as run by the Linear SVC implementation in Scikit-learn):

- 500 principal components
- L2 regularisation
- Regularisation parameter ‘C’: 0.01
- Squared hinge loss

The confusion matrix and output results are given below.

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	484	49	30	19	25	27	30	56	199	81
	automobile	66	501	13	22	18	27	40	59	83	171
	bird	120	57	220	64	107	85	171	93	51	32
	cat	65	78	69	179	46	196	176	57	54	80
	deer	66	33	99	49	259	85	199	141	29	40
	dog	48	73	73	118	72	333	100	79	65	39
	frog	29	60	55	70	71	76	527	50	25	37
	horse	51	58	51	48	65	66	43	479	44	95
	ship	138	75	10	16	9	36	17	21	569	109
	truck	68	187	15	18	9	21	49	49	101	483

Figure 17: SVM Confusion Matrix

	precision	recall	f1-score
airplane	0.4264	0.4840	0.4552
automobile	0.4278	0.5010	0.4644
bird	0.3465	0.2200	0.2832
cat	0.2968	0.1790	0.2379
deer	0.3803	0.2590	0.3197
dog	0.3498	0.3330	0.3414
frog	0.3898	0.5270	0.4584
horse	0.4419	0.4790	0.4604
ship	0.4664	0.5690	0.5177
truck	0.4139	0.4830	0.4484
avg/total	0.3940	0.4034	0.3987

Figure 18: SVM Summary Table

The results of the SVM model were slightly worse than the logistic regression, with an accuracy of 40.34%. As with LR, SVM had difficulty discriminating between the various types of animals, and the various transport objects. This suggests fundamental non-linearity in the structure of the underlying data that a linear model is unlikely to be able to resolve.

3 Non-linear models

3.1 K-Nearest Neighbours (KNN)

Given the structural difficulties in the data and the poor performance from the linear models, it was hoped that the non-linear aspect of nearest neighbour would provide some meaningful improvement. Grid search revealed the optimal parameters as:

- 20 principal components
- 20 neighbours

The results of the model are given below.

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	579	25	74	15	40	8	49	18	166	26
	automobile	64	408	48	33	60	23	103	20	144	97
	bird	127	15	375	49	174	35	149	33	29	14
	cat	64	22	122	202	127	111	229	51	31	41
	deer	71	14	225	28	402	12	157	50	33	8
	dog	49	13	140	126	116	280	159	61	38	18
	frog	21	12	154	29	151	25	565	22	12	9
	horse	71	28	103	48	157	42	103	367	39	42
	ship	133	53	27	31	36	11	21	17	634	37
	truck	81	122	40	31	41	15	88	53	146	383

Figure 19: KNN Confusion Matrix

	precision	recall	f1-score
airplane	0.4595	0.5790	0.5193
automobile	0.5730	0.4080	0.4905
bird	0.2867	0.3750	0.3308
cat	0.3412	0.2020	0.2716
deer	0.3083	0.4020	0.3551
dog	0.4982	0.2800	0.3891
frog	0.3481	0.5650	0.4566
horse	0.5303	0.3670	0.4487
ship	0.4984	0.6340	0.5662
truck	0.5674	0.3830	0.4752
avg/total	0.4411	0.4195	0.4303

Figure 20: KNN Results Summary

The results of the kNN model were very similar to the linear model performance, both in terms of accuracy and ability to distinguish between classes. On average the f-scores for 'sentient' classes improved over the linear models and kNN has an improved ability to predict between horses and deer, but still remains relatively poor.

3.2 Random Forest (RF)

The Nearest Neighbours algorithm can be considered a 'naïve' non-linear model, and the random forest a step up in complexity. Grid search revealed the below optimal hyperparameters:

- 100 principal components
- 200 estimators
- 25% of features at each node
- Maximum depth of 10

The results of the model are given in the following confusion matrix and performance statistics:

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	533	53	28	25	25	20	41	24	183	68
	automobile	30	591	11	24	10	24	26	28	79	177
	bird	143	50	144	73	190	73	175	57	49	46
	cat	57	58	47	214	50	191	175	61	47	100
	deer	68	24	56	46	391	55	220	63	38	39
	dog	39	47	60	146	60	356	121	82	38	51
	frog	15	33	53	40	135	53	571	26	28	46
	horse	45	61	24	48	102	86	64	390	45	135
	ship	103	98	8	18	10	35	24	19	594	91
	truck	40	206	6	22	13	29	38	34	82	530

Figure 21: RF Confusion Matrix

	precision	recall	f1-score
airplane	0.4967	0.5330	0.5149
automobile	0.4840	0.5910	0.5375
bird	0.3295	0.1440	0.2368
cat	0.3262	0.2140	0.2701
deer	0.3966	0.3910	0.3938
dog	0.3861	0.3560	0.3711
frog	0.3924	0.5710	0.4817
horse	0.4974	0.3900	0.4437
ship	0.5021	0.5940	0.5481
truck	0.4131	0.5300	0.4715
avg/total	0.4224	0.4314	0.4269

Figure 22: RF Results Summary

The random forest handles non-linearity in data better than linear models, due to the lack of assumptions on data structure. However, in this instance, the model was not able to meaningfully outperform the linear models at all, with an accuracy of 43.14%. In addition, the model misclassified many of the same objects as the other models, suggesting the underlying data is far more complex than these simple models are able to deal with effectively.

3.3 Gradient Boosting Machine (GBM)

The GBM is able to best capture the non-linear relationships found within the CIFAR-10 dataset for classifiers contained within the Scikit-learn package, with an accuracy of 51.32%. The below hyper-parameters were used for the GBM:

- 100 principal components
- 200 estimators
- 25% of features at each node
- Maximum depth of 10

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	596	40	58	38	19	19	18	29	126	57
	automobile	33	626	20	38	10	28	14	21	46	164
	bird	77	29	399	116	118	62	102	54	19	24
	cat	29	29	91	393	44	195	88	59	23	49
	deer	46	7	133	108	425	56	111	80	16	18
	dog	22	23	81	225	50	406	64	81	21	27
	frog	9	21	97	96	76	53	591	24	14	19
	horse	31	29	54	79	71	99	29	522	11	75
	ship	100	80	19	37	21	25	13	12	616	77
	truck	43	178	20	55	8	27	25	32	54	558

Figure 23: GBM Confusion Matrix

	precision	recall	f1-score
airplane	0.6045	0.5960	0.6002
automobile	0.5895	0.6260	0.6077
bird	0.4105	0.3990	0.4047
cat	0.3316	0.3930	0.3623
deer	0.5048	0.4250	0.4649
dog	0.4186	0.4060	0.4123
frog	0.5602	0.5910	0.5756
horse	0.5711	0.5220	0.5466
ship	0.6512	0.6160	0.6336
truck	0.5225	0.5580	0.5402
avg/total	0.5164	0.5132	0.5148

Figure 24: GBM Results Summary

As shown in the confusion matrix above, the GBM does not suffer from the same low bias for the bird, cat, deer and dog classes as was seen with earlier implementations.

4 Convolutional Neural Network (CNN)

As discussed in Chapter 4 a range of CNN models were tested, however Model Four was determined to be the best performing. The model achieved an accuracy of 82% with only 25 epochs of training, and a relatively short training time. The model did not use any image augmentation and little pre-processing. From both the Confusion Matrix and Summary Results table (Figure 265 and Figure 256), it was clear that the model was accurate at predicting most classes except for cat and dog. This is a well-documented issue for deep learning models, research indicates this problem can be overcome with ensembles of models that are focused on cat/dog classification. The CNN model outperformed all other models that were tested by at least 30%. Simpler machine learning models generally struggle on high dimensional data as their linear approach prevents fitting to such a large data variance, whereas deep learning models tend to succeed on such complex data as they can train and adjust to the range of data points.

		Predict									
TRUE	Class	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
	airplane	841	7	50	15	14	4	5	4	33	27
	automobile	11	893	1	7	1	1	4	1	12	69
	bird	43	2	737	46	46	49	41	22	8	6
	cat	15	4	53	646	46	140	51	22	14	9
	deer	14	0	42	45	830	26	16	21	4	2
	dog	11	3	26	114	30	765	9	33	3	6
	frog	7	1	34	41	19	14	875	5	2	2
	horse	11	0	14	41	46	37	5	832	1	13
	ship	53	19	13	7	1	2	2	1	882	20
	truck	15	32	5	13	2	5	2	5	16	905

Figure 25: CNN Confusion Matrix

	precision	recall	f1-score
airplane	0.82	0.84	0.83
automobile	0.93	0.89	0.91
bird	0.76	0.74	0.75
cat	0.66	0.65	0.65
deer	0.80	0.83	0.82
dog	0.73	0.77	0.75
frog	0.87	0.88	0.87
horse	0.88	0.83	0.86
ship	0.90	0.88	0.89
truck	0.85	0.91	0.88
avg/total	0.82	0.82	0.82

Figure 26: CNN Results Summary

5 Discussion

It was expected at the outset of this experiment that the pre-CNN models would not perform as well as the CNN model. However, the fact that the discrepancy was so significant with comparatively little tuning of the CNN was unexpected.

In addition, the non-linear models could not significantly outperform the linear models, which was a further unexpected outcome. It became apparent that the similarities and complexities within the data set were beyond the range of these simple models entirely, and that the CNN was much more suited to attacking a problem of this nature.

There were several classes that were commonly confused with each other. In particular, airplanes were confused with birds and ships, trucks were confused with automobiles, and the various animals were frequently confused with each other. This inability to distinguish between these classes persisted across the pre-CNN models (and partially appeared in the CNN model between cats and dogs as well), suggesting these objects have fundamental similarities that a simple model cannot isolate.

Chapter 6: Personal Reflection

The work load of this assignment was split between the three team members, each working on a different aspect. Therefore, each individual has a different perspective and insight gained from undertaking this research. This chapter discusses each member's personal reflection.

1 Reflection – Cameron Wasilewsky

Undertaking the CNN model development provided not only practical experience with deep learning image classification but conveyed its true power. Although it was initially difficult to set up the first CNN and get it running on a cloud server, once run it was easy to manipulate, tune and build upon. It was not much harder than implementing a linear model, yet its performance was exponentially better. Thus, this research emphasised the future growth and potential of machine learning.

2 Reflection - Kristopher Lopez

Despite the extensive grid search performed across pre-CNN models, none were able to represent the complex relationships throughout the dataset as well as the CNNs. All linear models and naïve non-linear models such as KNN were only able to achieve up to 42% accuracy whilst the increased non-linearity and flexibility of the Random Forest ensemble moved the needle to close to 45% accuracy and Gradient Boosting Machines up to 51.32%. Other research papers indicate the usefulness of a KNN ensembled with CNN, and the grid search used in this paper would be a great platform to explore whether a multi-layered ensemble of pre-CNN models and CNN models could improve performance.

3 Reflection - Arjun Sathasivam

Both linear and non-linear pre-CNN models found it difficult to separate airplanes and ships, automobiles and trucks, and performed poorly on living creatures, birds, cats, dogs, frogs and horses. It is interesting to note that the models found it difficult to distinguish between objects that humans would also consider similar to each other, suggesting the learning behaviour of humans is not far removed from the statistical methods employed here.

Chapter 7: Conclusions and Future Work

1 Conclusion

Table 2: Summary Results, all models

Classifier	Accuracy	Precision	Recall	F-Score
RF	36.14%	37.37%	36.14%	36.28%
5-kNN w/ PCA	39.24%	41.48%	39.24%	39.39%
SVM w/ PCA + Tuning	40.34%	43.67%	40.34%	41.45%
LR	40.61%	41.57%	40.61%	40.97%
LR + Tuning	40.87%	41.93%	40.87%	41.25%
LR w/ PCA + Tuning	41.74%	42.72%	41.74%	42.10%
20-kNN w/ PCA	41.95%	45.48%	41.95%	42.35%
RF w/ PCA Whitening + Tuning	43.04%	45.73%	43.04%	43.80%
RF w/ PCA + Tuning	43.64%	47.94%	43.64%	44.98%
GBM w/ PCA + Tuning	51.32%	51.37%	51.32%	51.25%
CNN (96, 192, 384)	79.88%	80.07%	79.88%	79.97%
CNN (32, 64, 128)	80.32%	80.25%	80.32%	80.28%
CNN (48, 96, 192)	82.06%	82.12%	82.06%	82.09%

Following an exhaustive grid search, four candidate models were selected and optimised to perform the image recognition classification task captured by the CIFAR-10 data set. In addition, a convolutional neural network was explored as a means of obtaining better performance (as guided by current research). The results of this can be seen in Table 2.

As expected, the CNN model performed extremely well when compared to the simpler linear and non-linear models selected to tackle this image recognition task. The difference in performance was significant, suggesting the CNN is by far the preferred tool for this type of problem.

2 Future Work

As this was a very clean and straight forward dataset there is a large potential for future work as it is easy to work with. Going forward development and testing new models can be done on the data set or alternatively one can take the models developed and apply them to more complex problems.

2.1 Other Models Same Data

At the forefront of machine learning visual analysis is Generative Adversarial Networks (GAN), Restricted Boltzmann Machines, Residual networks and Residual Networks (ResNets) models, which could be applied to this problem given more time and skill.

There was limited opportunity to investigate feature reduction methods for the pre-CNN models beyond PCA. There are several techniques in common use, including whitening and mirroring that could be explored in the future. In addition, data augmentation is a frequently used technique to improve the performance of CNN models. This involves adding features by translating, mirroring and cropping sections of images before training the CNN. In future work this technique could be explored further.

2.2 Other Data Similar Models

The best performing CNN model of identification could easily be adapted and applied to a range of other problems such as classifying more objects (CIFAR100). The concepts can also be applied to security screening, facial recognition and self-driving cars. With the advent and improvements in aerial photography from both drones and satellite; large scale projects could be undertaken. For example, animal tracking or rescue in natural disasters. With satellite photos, environmental changes, human migration and movement, and surveillance can all be automatically monitored.

Bibliography

[1] D. Cresan, U. Meier and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," *Galleria 2,* pp. 1-8, 2012.

[2] A. Krizhevsky, "The CIFAR-10 Dataset," CS.Toronto.edu, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed 30 May 2017].

[3] Y. Abouelnga, O. Ali and H. Rady, "CIFAR-10: KNN-based Ensemble of Classifiers," in *International Conference on Computational Science and Computational Intelligence*, Las Vegas, 2016.

[4] L. Wan, M. Zeiler, S. Zhang, Y. LeCun and R. Fergus, "Regularization of Neural Networks using DropConnect," CS.NYU, 16 December 2013. [Online]. Available: http://cs.nyu.edu/~wanli/dropc/. [Accessed 30 May 2017].

[5] Y. LeCun, C. Cortes and C. Burges, "The MNIST Database of Handwritten Digits," Lecun, 2012. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 30 May 2017].

[6] M. Liang and X. Hu, "Recurrent Convolutional Neural Network for Object Recognition," *CVF,* vol. 3, no. 2, pp. 3367-3375, 2014.

[7] ScikitLearn, "Machine Learning Models," ScikitLearn, 2016. [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html. [Accessed 30 May 2017].

[8] F. Chollet, "How convolutional neural networks see the world," Keras Blog, 30 January 201. [Online]. Available: https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html. [Accessed 28 May 2017].

[9] B. Graham, "Fractional Max-Pooling," *Computer Vision and Pattern Recognition,* vol. 2, no. 1, pp. 15-20, 2014.

[10] A. Karpathy, "Lessons Learned from Manually Classifying CIFAR-10," Hacker's Guide to Neural Networks, 27 April 2011. [Online]. Available: http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/. [Accessed 30 May 2017].

[11] J. Brownlee, "Display Deep Learning Model Training History in Keras," Machine Learning Mastery, 17 June 2016. [Online]. Available: http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/. [Accessed 28 May 2017].

Appendix

1 Instructions on how to run code for pre-CNN models

1.1 Required Packages

- Numpy
- Scipy
- Matplotlib
- Time
- CSV
- Xlsxwriter
- Sys
- OS
- Keras
- Tensorflow

1.2 Recommend Computer Specifications

Table 3: Computer specifications

Processor	RAM	System Type	Operating System	Python Version
Intel(® Core™ i702600 CPU @ 3.40GHz	16.00GB	64-Bit x64-based processor	Windows 10 Home	3.5

1.3 Steps

1. Unzip provided file
2. Navigate to '…\COMP5318_Assignment_1_Group_34\code\algorithm'
3. Hold 'shift' and right click in the folder directory
4. Select "Open command window here"
5. Enter 'python main.py' into the command window and press enter
6. You will be prompted to select which model you wish to run. The options are outlined in the table below

Table 4: Model Options

Option	Model Name	Description	Classifiers	Expected Accuracy	Expected Run Time
1	SGD Classifier	**Components**: <No PCA used> **Loss:** Modified Huber **Regularisation**: 0.5 x L2	SGD	0.3721	0.5 (m)
2	Linear SVC	**Components**: 500 **Loss:** Squared Hinge **Regularisation**: 0.01 x L2	SVM	0.4034	3.5 (m)
3	Logistic Regression	**Components**: 1000 **Regularisation**: 0.01 x L2	LR	0.4174	2.5 (m)
4	Nearest Neighbours	**Components**: 20 **Neighbours**: 20	kNN	0.4195	4.0 (m)
5	Random Forest	**Components**: 100 **Trees**: 200 **Depth of tree**: 10 **Features per tree**: 25%	RF	0.4314	3.5 (m)
6	Gradient Boosting	**Components**: 100 **Trees**: 200 **Depth of tree**: 10 **Features per tree**: 25%	GBM	0.5132	50 (m)
7	AdaBoost	**Components**: 100 **Estimators:** 200	AB	0.3701	5 (m)
8	Ridge Classifier	**Components**: 1000 **Regularisation**: 0.01 x L2	RC	0.3964	2.5 (m)
9	Extra Trees	**Components**: 100 **Trees**: 200 **Depth of tree**: 10 **Features per tree**: 25%	ET	0.4343	2.5 (m)

2 Instructions on how to Run CNN Code

2.1 Required Packages

Required to install Floyd

2.2 Recommend Computer Specifications

Table 5: Computer specifications

Processor	RAM	System Type	Operating System	Python Version
GPU	12.00GB	Floyd Server	Windows	3.5

2.3 Steps

1. Go to www.floydhub.com
2. Create a *'floydhub'* account
3. On your machine in the command prompt type in `pip install -U floyd-cli`
4. Login to your account with `floyd login`
5. Press 'y'
6. Copy and paste your authentication token with 'ctrl + c', 'ctrl + v', if this does not work, use the right click button to do so.
7. Navigate to '…\COMP5318_Assignment_1_Group_34\code\algorithm'
8. Shift, right click and open a command window at this location
9. Initialise the Floyd project `floyd init cnntest`
10. To run the code on the server enter `floyd run --env tensorflow --gpu "python CNN.py"`

You can track the process with the log provided or go to Floyd and go to experiments to track the progress and view the log

3 Pre-CNN Model Background and Additional Results

3.1 Logistic Regression

The logistic regression is a linear classification tool, specified as

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \sigma(\eta)^{y}\big(1-\sigma(\eta)\big)^{1-y},$$

where $\eta = \boldsymbol{w}^T\boldsymbol{x}$ and $\sigma(\eta)$ is the sigmoid function $\frac{1}{1+e^{-\eta}}$.

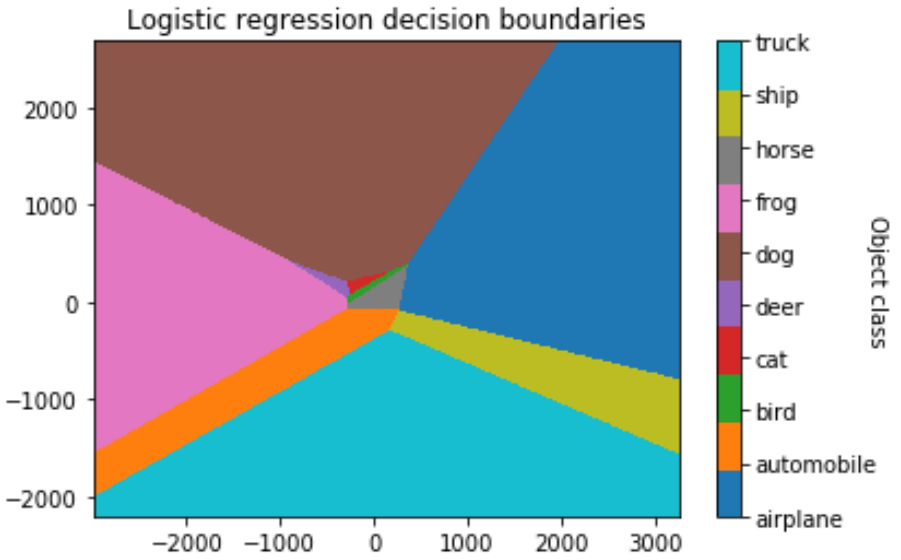
Optimising the logistic regression function reduces to minimising the maximising the conditional likelihood W, which can be derived as

$$W = \max_{W} \prod_i P(Y_i|X_i, W).$$

As with other linear regression models, the conditional likelihood is not optimised directly. Instead, the conditional log likelihood (with optional regularisation parameter λ) is maximised, given by

$$W = \max_{W} \sum_i \ln P(Y_i|X_i, W) - \frac{\lambda}{2}\big||W|\big|^2.$$

The logistic regression produced a set of decision boundaries for each of the classes in the CIFAR-10 data set. These are shown graphically below (using optimised hyper-parameters and a PCA reduction to two dimensions):



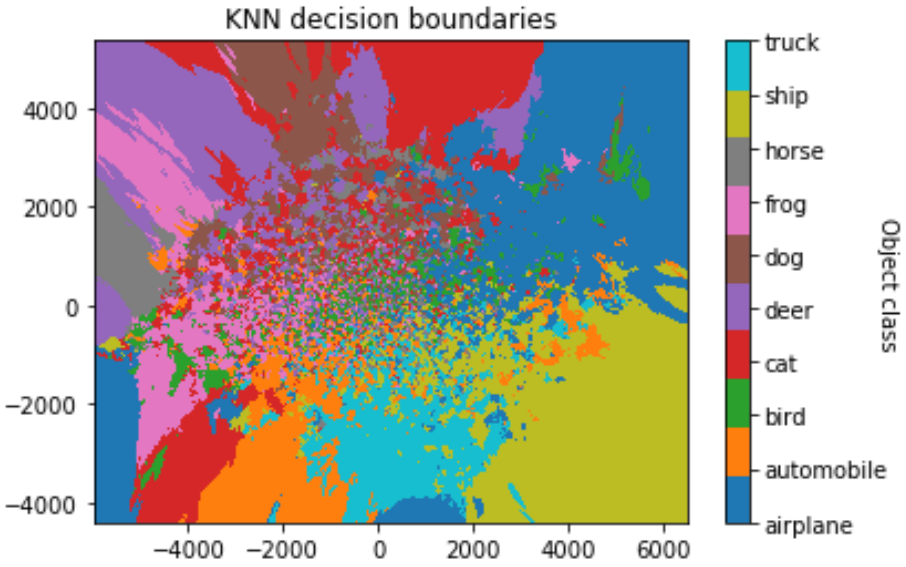
3.2 K-Nearest Neighbours

The k-nearest neighbours algorithm works by identifying the nearest k training points from the given test point, and selecting the most popular class of its neighbours as the predicted class. The algorithm is simple, but computationally expensive, and performs poorly in high dimensional spaces.

The distance function can be set as a parameter to the model, however the most common choice is Euclidean distance, defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_i (x_i - y_i)^2}.$$

Using an optimised k-value of 20, the decision boundaries for the KNN model are shown below (with a PCA reduction to two dimensions only). Each of the 10 classes are represented by a separate colour in the plot.



3.3 Random Forest

The random forest model is an ensemble learning tool, which takes the result of many individual decision trees and combines them to produce a single prediction.

Each tree is generated by taking a random selection of features at each node of the learning process, until the desired depth of tree is reached. To derive a predicted class from the multitude of trees created through this process, a majority vote is taken from the tree output.

3.4 Gradient Boosting

As with the random forest, gradient boosting is an ensemble method, designed to combine the results of many weak learners into a single more powerful prediction. In particular, the model is designed to gradually improve by iteratively fitting learners to residuals (usually using decision trees).

The gradient boosting function is given by:

$$F(x) = \sum_{i=1}^{N} \gamma_i h_i(x) + c$$

where h are the individual weak learners, and γ are the relative weights assigned to each learner.

Each γ is chosen to minimise the chosen loss function L at each iteration, and can be found at iteration m to be:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

where F_m is the gradient boosting function at iteration m.

The loss function chosen for this analysis was the 'deviance' loss function, which is equivalent to a logistic regression loss.

4 CNN Model Components

- Convolution2D: Convolves the input of small selected sections of the image for two dimensions. Simply put a sliding window moves across the image, multiplying the matrix throughout deepening the building block

- MaxPooling: Reduces the dimensions of the convolutional layer, pooling together values. This is achieved by taking the max value of the area being pooled

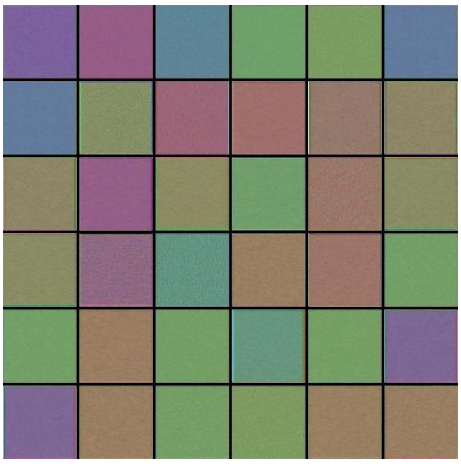
- Drop out: A drop out layer is used to prevent overfitting. Rather than becoming reliant on certain node paths with each iteration certain nodes are ‘dropped out’.

- Flatten: Prior to being flattened the network is in multiple dimensions, to allow for classification it needs to be flattened into one dimension so it can be used in the Dense layer.

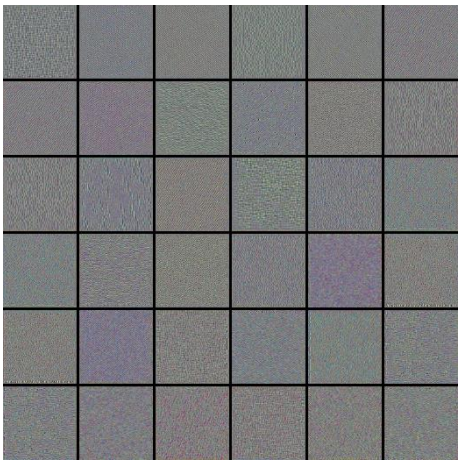
- Dense: A simple ANN classifier used to identify which class the final image should fall into

5 CNN Layer Visualisation [8]

5.1 Initial Layer



5.2 Middle Layer



5.3 Final Layer