Course code and name:	F20DL: Data Mining and Machine Learning
Type of assessment:	Group 11
Coursework Title:	Data Analysis and Bayes Nets Part 4
Student Names:	Hana Khan, Neil Patrao, Yasir Rayamarakar Veetil, George Chandy, Fatima Patel
Student ID Numbers:	H00379151, H00385850, H00385488, H00377409, H00339652

Declaration of authorship. By signing and uploading this form:

☐ **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

☐ I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

☐ I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature: *Hana Khan, Neil Patrao, Yasir Rayamarakar Veetil, George Chandy, Fatima Patel*
Date: *27/11/2023*

Data Mining and Machine Learning

Neural Networks and Convolutional Neural Networks

F20DL CW PART FOUR

Group11 {Hana Khan, Neil Patrao, Yasir Rayamarakar Veetil, George Chandy, Fatima Patel}

Stop Sign Dataset Notebook:

https://colab.research.google.com/drive/142K1EdFiSu2mTBqErcNizAaK9BQUGI7c?usp=sharing

Preprocessing

Normalizing

* The features in the given dataset are between 0 and 255. Due to this high numerical scale, the model's learning process can be skewed, leading to suboptimal performance. Normalization ensures each feature contributes proportionately to the model by giving them the same scale.

Gaussian Blur with Binary Adaptive Thresholding

* Gaussian Blur helps smoothing the images, reducing noise and irrelevant details making it easier for the model to identify relevant patterns.
* Binary Adaptive Thresholding further simplifies the image by focusing on important features, which is particularly effective in handling varied lighting conditions in traffic sign images.
* The combination of blurring and thresholding enhances the contrast between the signs and their backgrounds, making it easier for the models to distinguish between different classes. This is especially beneficial for the classes that have subtle differences.

Logistic Regression (Linear Classification)

Model	Dataset Model Trained on	Train Accuracy	Test Accuracy	Difference	Mean Accuracy
LogisticRegression	X_train normalized	0.972	0.899	0.072	N/A
LogisticRegression with L2 penalty (C = 0.1)		0.941	0.884	0.058	
LogisticRegression with L1 penalty (C = 0.1)		0.890	0.87	0.021	
10-Fold CV		N/A			0.885
Stratified 10-Fold CV					0.941
LogisticRegression	Gaussian Blur with Binary Adaptive Thresholding on X_train then normalized	0.999	0.857	0.143	N/A
LogisticRegression with L2 penalty (C = 0.1)		0.841	0.783	0.058	
LogisticRegression with L1 penalty (C = 0.1)		0.791	0.782	0.009	

10-Fold Cross Validation

* 10-fold CV distributes the dataset into ten subsets, training the model on nine subsets and validating on the remaining one repeatedly, ensuring comprehensive assessment of the model's performance across the entire dataset and reducing the risk of bias from any data partitioning. The accuracy was relatively high at 88.5%

Stratified 10-Fold Validation

* The accuracy increased by 5.6% compared to using regular cross-validation. This is because stratified Cross-Fold validation ensures that each fold maintains the same proportion of each class as in the original dataset. Regular cross-validation might result in some folds not containing any samples of the minority class, leading to a skewed understanding of the model's performance.
* The 10-Fold Stratified Cross-Fold validation's superior performance demonstrates its effectiveness in dealing with unbalanced datasets. It ensures that each fold is representative of the overall dataset, thereby providing a more accurate and consistent evaluation of the model's ability to generalize across all classes. This method slightly addresses the overfitting issue observed in regular 10-Fold Cross-Fold validation by offering a more robust and realistic assessment of the model's performance.

How well does the linear classifier generalize to new data?

* The model's generalization can be inferred from the difference between the training and test accuracy. A smaller difference suggests better generalization.
* Training logistic regression on x_train normalized without CV or regularization gave a difference of 7.2% in accuracy which suggests overfitting. Using L2 and L1 regularization, the difference reduced to 5.8% and 2.1% This indicates the model has relatively good generalization,
* Using preprocessing and regularization further increased the model's ability to generalize. The least difference between the test and train accuracies was 0.9% which was for LogisticRegression with L1 penalty (C = 0.1). This is because L1 regularization promotes model simplicity and reduces overfitting by enforcing sparsity in the feature weights.

What hypothesis can you make about this data set being linearly separable or not?

* If a linear classifier achieves high accuracy on both the training and test sets, it often indicates that the data is linearly separable or close to it. However, if the test accuracy is significantly lower than the training accuracy, it may suggest overfitting or that the data is not perfectly linearly separable.
* Given the high accuracies achieved by the models, especially with stratified cross-validation, the dataset is almost linearly separable with 94% accuracy.
* The slightly lower accuracies for the regularized models suggests that while the dataset is largely linearly separable, some of the features may not contribute significantly to the classification or may introduce noise. Regularization helps in mitigating the impact of such non-informative features, leading to a model that generalizes better but with a slight compromise on training accuracy to improve performance on unseen data.

Multilayer Perceptron

Experimenting with various neural network parameters Manually

Note: The layers displayed in the table only represent the hidden layers. The ANN models utilized in these experiments had an input layer consisting of neurons equal to the total pixel count in the image (in this case, 2304 pixels) and an output layer consisting of neurons equal to the number of classes (in this case, 10). Softmax was used as the activation function for the output layer.

Experiment No.	Hidden Layer Number	Activation Functions	Number of Neurons	Learning Rate	Number of Epochs	Momentum	Validation Threshold	Test Accuracy	Test Loss
1	1	Tanh	300	0.01	100	0.9	0.3	62.8%	2.47
	2	Tanh	100						
2	1	Tanh	256	0.01	30	0.9	0.3	66.7%	1.76
	2	Tanh	128						
	3	Tanh	64						
3	1	ReLU	300	0.01	30	0.9	0.3	56.5%	1.93
	2	ReLU	100						
4	1	Tanh	256	0.001	30	0.9	0.1	65.5%	1.24
	2	Tanh	128						
	3	Tanh	64						
5	1	Tanh	256	0.01	30	0.9	0.1	61.5%	1.37
	2	Tanh	128						
	3	Tanh	64						
6	1	Tanh	256	0.01	30	0.4	0.1	60.9%	1.33
	2	Tanh	128						
	3	Tanh	64						

Experiment results:

* Experiment 1-3 was done by varying the number of layers, the activation functions, and the number of neurons in each layer while keeping the other parameters constant. We see that a MLP of 3 layers with 256, 128 and 64 neurons in each layer respectively and Tanh as activation function for each layer gave the most test accuracy of 66.7%.
* Experiment 4-6 was done by keeping the values optimal values from the results of the previous experiments, i.e., the number of layers, the activation functions, and the number of neurons in each layer, constant and varying the learning rate, momentum, and validation threshold. We see that learning rate, momentum, and validation threshold values of 0.001, 0.9 and 0.1 gave the most test accuracy of 65.5%.
* Manually tuning hyperparameters often leads to inconsistent model performance due to the sensitivity of the model to those settings. Each training run with fixed hyperparameters might yield different accuracies due to several factors:

1. Random Initialization: Neural networks initialize weights randomly, leading to different starting points for optimization.
2. Optimization Sensitivity: Even slight variations in hyperparameters can significantly influence the optimization path.

Experimenting with various neural network parameters using Keras Random Search

Random search was used to randomly pick combinations of hyper parameters from the following range of hyper parameters:

Hyperparameter	Range or Options	Reason	Practical Consideration
Number of Layers	1 to 5	Neural network depth impacts its ability to model complex relationships. A range of 1 to 5 allows for experimenting with both shallow and deeper architectures, suitable for understanding whether the problem requires simple or complex representations.	Too many layers may lead to overfitting and increase computational cost.
Neurons per Layer	32 to 512 (step of 32)	The number of neurons in a layer determines the representational capacity of that layer. Ranging from 32 to 512 provides a wide spectrum from a relatively simpler to a more complex model.	Fewer neurons may not capture complex patterns well, whereas too many can cause overfitting and increased computational load.
Activation Function per Layer	'relu', 'tanh'	'relu' is commonly used for its efficiency and effectiveness in avoiding the vanishing gradient problem, while 'tanh' can sometimes perform better for certain types of data, especially where scaling of input features is important.	Including both allows assessing which activation function works best for the specific nature of the dataset.
Learning Rate	1e-4 to 1e-2 (logarithmic sampling)	The learning rate controls how much to update the model in response to the estimated error. A logarithmic range allows exploring both subtle and more significant weight updates to find a balance between convergence speed and stability.	Too high a learning rate can cause the model to converge too quickly to a suboptimal solution, or diverge, while too low a rate can make training unnecessarily long and prone to getting stuck in local minima.

Momentum	0.5 to 1.9 (step 0.1)	Momentum helps to accelerate the optimizer in the relevant direction and dampens oscillations. This range allows experimenting from no momentum to high momentum.	Properly tuned momentum speed up training and escape local minima, but too much momentum might overshoot the minima.
Number of Epochs	Up to 30	This determines how many times the model will see the entire training dataset. A range from 10 to 50 allows the model to learn from the data adequately without excessive overfitting.	Fewer epochs might not be sufficient for the model to learn from the data, whereas too many epochs can lead to overfitting and increased training time.
Validation Threshold	0.2 – 0.6 (step of 0.1)	Range 0.2 to 0.6 allows for sufficient data to validate the model while ensuring enough data remains for training. Helps in assessing model's performance on unseen data and in preventing overfitting.	Choosing very small split can lead to overfitting. Choosing too large split reduces amount of data for training, which can degrade model performance due to insufficient training. The right balance is crucial for model generalization.

- For the optimum number of epochs, instead of running the models' multiple times with different number of epochs, the accuracies were printed for each epoch and the best value was chosen when the accuracies did not increase after that number of epochs. This was at 20 epochs.
- For the best validation split, the accuracy of validation slightly decreased as the value of the split increased, the best split was 0.2 with 0.895% validation accuracy.

The best model:

Best val_accuracy: 0.8928802609443665
Model: "sequential"
Test Set Accuracy: 0.89
Best hyperparameters:
- Number of hidden layers: 3
- Units in layer 0: 96
- Activation function for layer 0: tanh
- Units in layer 1: 160
- Activation function for layer 1: tanh
- Units in layer 2: 192
- Activation function for layer 2: tanh
- Learning rate for the optimizer: 0.003814120220610665
- Momentum for the optimizer: 0.7

Training Set Accuracy: 0.96
Training Set loss: 0.13
Test Set Accuracy: 0.89
Test Set Loss: 0.52

CNN

The best validation score was 94% and the best hyper parameters are reflected in this table below alongside the ranges of values tested:

Hyper Parameter	Range	Best	Reason
Conv1 Filters	[32, 128] step of 32	32	A range starting from 32 filters allows the model to start learning from a reasonable feature space, increasing to 128 to allow the model to capture more complex features. The step of 32 is chosen to provide a balance between model complexity and computational efficiency.
Conv1 Kernel Size	[3, 5]	4	Smaller kernels are generally better for capturing fine-grained details, while larger kernels can capture more abstract features. The range [3, 5] is a common choice that allows the network to learn spatial hierarchies effectively.
Conv2 Filters	[32, 128] step of 32	32	Like Conv1, this range allows for an increase in the complexity of learned features in deeper layers without a large computational cost. The progressive increase in filters often corresponds to a model's increased capacity to represent more abstract features.

Conv2 Kernel Size	[3, 5]	3	Maintaining the same range for the second convolutional layer allows for consistency in the size of the learned spatial features. It's common to keep the kernel sizes the same or increase slightly in deeper layers.
Number of Dense Layers	[1, 3]	3	The number of dense layers is chosen to experiment with different levels of abstraction in the classification task. A single layer may be sufficient for simple problems, while more layers can create a deeper hierarchy of concepts for more complex tasks.
Number of neurons in dense layer 0	32 to 512 step of 32	256	*Same reasons as above for MLP*
Number of neurons in dense layer 1		64	
Number of neurons in dense layer 2		192	
Learning rate	1e-4 to 1e-2 (logarithmic sampling)	0.0028438	
Momentum	0.5 to 1.9 (step 0.1)	0.9	
Dense Activation	'relu', 'tanh'	tanh	

Final Conclusions

- The data is linearly separable since the accuracies for the train and test set are high, 96% and 89% respectively. Additionally, deep neural networks can generalize to new data after tuning the parameters and regularizing the models. Regularized logistic regression on the pre-processed normalized dataset gave the smallest difference in accuracies of train and test of 0.9%
- When the MLP had 1 layer, relu outperformed tanh but when the network had 3 layers, tanh worked better. This could be because in networks with more layers, tanh tends to normalize the data (due to its output range), which can lead to better convergence properties. The best MLP had 3 layers with tanh in each layer and softmax for the output.

Name	Contributions
Neil Patrao	Cross validation, Stratified Cross validation
Hana Khan	Ran linear classifier on train set and noted the accuracies on test set in report, Built and trained multilayer perceptron and tuned manually hyperparameters, Experimented with various hyperparameter values manually and noted the best parameter values in report.
George Chandy	Built and trained multilayer perceptron, Made CNN and tuned hyper parameters using random search
Yasir	Data preprocessing
Fatima Patel	Linear regression regularization, Linear regression on preprocessed data, Linear regression report section, MLP cross validation hyper parameter tuning and report section, CNN report section