An Investigation of Classification Methods for CIFAR-10

Natalie Dobrinska 20843427
Melissa Kang 26368951
Ashish Ghimire 88116593

Summary

This report outlines the process of fitting the k-Nearest-Neighbors classifier, logistic classifier, feedforward neural network, and convolutional neural network on the CIFAR-10 dataset. The convolutional neural network with dropout performed the best on the pre-decided test dataset with an accuracy of 82.73%.

Data Description

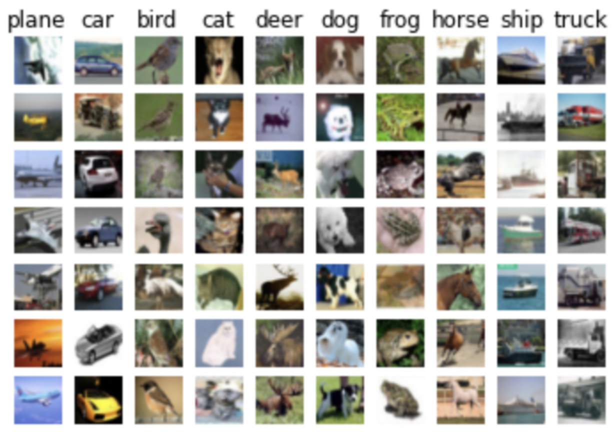


Figure 1. Visualization of 7 samples of each class in CIFAR-10 dataset.

The dataset used in this project is the CIFAR-10 dataset. This dataset contains 50,000 training and 10,000 pre-specified testing 32x3[1]2 colored images. Each image in the dataset is labeled with one of the ten corresponding classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck (Figure 1.). The classes are mutually exclusive and do not overlap. Each class has 5,000 images in the training data set.

The paper “Progressive neural architecture search” by Liu, Chenxi, et al. is a published paper that has used the CIFAR-10 dataset for research. The article introduces an innovative approach for learning the structure of convolutional neural networks that proves to be significantly more efficient than current advanced-level reinforcement learning algorithms. Liu, Chenxi, et al. explains that their approach uses a sequential model-based optimization strategy where the structure space is searched in increasing complexity while at the same time a surrogate model is being trained to model the search through the space. Their research concluded that under the same conditions, the method was up to 5 times more efficient than the models evaluated using modern reinforcement learning methods in Zoph et al. (2018) and up to 8 times in terms of computation time. The structures identified through this approach gave way to advanced classification accuracies on the CIFAR-10 dataset.

[1] Liu, Chenxi, et al. "Progressive neural architecture search." *Proceedings of the European conference on computer vision (ECCV)*. 2018.

Classifiers

k-Nearest-Neighbors Classifier (kNN): The kNN classifier takes the labels of the nearest k datapoints in the training dataset and classifies a new datapoint based on the majority label. We used Scikit-learn to train our classifier with hyperparameters n_neighbors, and the rest default. We investigated accuracies with different n_neighbors values ranging [1, 5, 10, 50, 100].

Multinomial Logistic Classifier: The multinomial logistic classifier is an extension of the logistic classifier in that it is used for data with more than two class labels. A logistic classifier is linear and produces only one set of weights (single template) for each of the classes. We used Scikit-learn to train our neural network with hyperparameters C, penalty, solver, max iterations, and fit intercept. We investigated accuracies with different inverse regularization (C) strength values ranging [0, 0.1, 1, 10, 50].

Feedforward Neural Network: The feedforward neural networks consist of layers of matrix-vector operations with non-linearities at each hidden layer, learning multiple templates per hidden unit. The object of interest, however, needs to be centered in the images. We used Scikit-learn to train our neural network with hyperparameters hidden_layer_sizes, activation, learning rate initial, solver, n_iter_no_change, max iterations, alpha, and batch size. We investigated accuracies with different hidden_layer_sizes values ranging [(256, 128, 64), (64,), (128, 64), (64, 32, 16)].

Convolutional Neural Network: The convolutional neural network takes small patch sizes of an image and the filter computes a weighted sum of pixels in each patch to create a hidden unit value. Each layer consists of filters, max-pooling, and nonlinearity, which gets flattened to create a fully connected feedforward neural network. The hyperparameters used for Conv2D were filters, kernel size, activation, kernel_initializer, padding, input_shape; for MaxPooling2D was pool_size, for Dropout was rate, for Dense were units, activation, kernel_initializer and to compile were optimizer and loss.

Experimental Setup

Use seed = 1234 for reproducibility.
For the kNN, multinomial logistic classifier, and feedforward neural network, classification accuracy and confusion matrices were used to compare the performance of models with different hyperparameters. Learning curves of the cross entropy loss and classification accuracy were used to evaluate the convolutional neural networks.
The CIFAR-10 dataset has a pre-specified test dataset that was used for the final evaluation. Only the models with the highest accuracy in their category were tested on the final dataset. The rest of the data, batches 1-5, was split into training and validation datasets with sizes 80% and 20% respectively.

```
kNN Classifier
k = [1, 5, 10, 50, 100]
For each k:
        Create kNN model with n_neighbors parameter = k
        Fit model on X train and y train datasets, predict labels for X train, predict labels for X test
        Get accuracy score for train and test
        Plot confusion matrix for differences between y test and X test predictions
```

Psuedocode for kNN hyperparameter tuning (Homework 1)

Multinomial Logistic Classifier

```
Inverse regularization values: [0, 0.1, 1, 10, 50]
For each c:
    Create logistic regression model with parameters C = c,
        penalty = l1 if c != 0 else none, solver = saga, max iterations = 300,
        And fit intercept = True
    Fit model on X train and y train datasets, predict labels for X train, predict labels for X test
    Get accuracy score for train and test
    Plot confusion matrix for differences between y test and X test predictions
```

Psuedocode for Multinomial Logistic hyperparameter tuning (Homework 2)

<u>Feedforward Neural Network</u>

```
Layer set =  [(256, 128, 64), (64,), (128, 64), (64, 32, 16)]
For each layer set:
    Create MLP classifier model with parameters hidden_layer_sizes = layer set,
        Activation = relu, learning rate initial = 0.01, solver = adam,
        n_iter_no_change = 150, max iterations = 100, alpha - 0.001,
        And batch size = 200
    Fit model on transformed X train and y train datasets, predict labels for transformed X test
    Get accuracy score for train and test
    Plot confusion matrix for differences between y test and X test predictions
```

Pseudocode for Feedforward Neural Network hyperparameter tuning (Homework 3)

<u>Convolutional Neural Network</u>

```
d = [None, 0.20, 0.25, 0.30]
For each d:
    Create model layer-by-layer using Sequential():
    Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
    padding='same', input_shape=(32, 32, 3)),
    Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform',
    padding='same')
    MaxPooling2D(2, 2)
    Dropout(d)
    2x (Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform',
    padding='same'))
    MaxPooling2D(2, 2)
    Dropout(d)
     2x (Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform',
    padding='same'))
    MaxPooling2D(2, 2)
    Dropout(d)
    Flatten()
    Dense(128, activation='relu', kernel_initializer='he_uniform')
    Dropout(d)
    Dense(10, activation='softmax')
    Compile model with optimizer=opt, loss='categorical_crossentropy',
    metrics=['accuracy']
    Get accuracy score X test predictions to y test
    Get plot of Cross Entropy Loss and Classification Accuracy
```

Pseudocode for Convolutional Neural Network hyperparameter tuning (https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/)

Experimental Results

Model type	Tuned hyperparameter	Test accuracy	Other Metric
kNN	k = 1	0.10	See appendix figure 6
Logistic	C = 0.1	0.10	See appendix figure 7
Feedforward Neural Network	hidden_layer_sizes = (128, 64)	0.4251	See appendix figure 8
Convolutional Neural Network	d = 0.25	0.8273	See appendix figure 9

Table 2. Results of Models on Final Test Dataset. Models were chosen based on preliminary results (see Appendix Table 1).

The conventional neural network has the highest test accuracy compared to the other models tested.

Insights

The most confused numeric label pairs across all of the classifiers on the test dataset were (3,5) and (8,0). The corresponding labels for 0, 3, 5, and 8 in that order are airplane, cat, dog, and ship. Cats and dogs could be confused by humans as well if the pictures are blurry enough, and it might also depend on the breed since some breeds of dogs and cats have more similarities than others.

The results on the test dataset were very surprising for the kNN and logistic classifiers since both classifiers only predicted one label. Since the test dataset has 1000 images per label, the accuracy for both was 10%. We were expecting the accuracy for both of these classifiers to be low, but we did not expect it to predict only class for every image in the test dataset.

The results from the convolutional neural network came back as expected and we were happy to see it predicted the test dataset with 82.73% accuracy. We were worried about the training data overfitting, but that does not seem to be the case with the relatively high accuracy.

One weakness of the classifiers examined in this report on a more realistic dataset would be that they are only able to predict one label per image, but most images have multiple objects in them. This puts a limitation on the accuracy of the classifiers because they are only able to correctly identify labels with one object in the image frame. For example, if any of the classifiers were to be used to tell if humans are in front of a car for a self-driving system, they might only predict that humans are in the frame if they look like the ones in the training dataset or if there is only one human in the frame. Additionally, it might mistake children to be inanimate objects.

A weakness of the feedforward neural network is that the object has to be centered in the image in order to predict the object more accurately. In contrast, a strength of convolutional neural network over the other methods is that it is able to learn basic visual features and can therefore be composed to recognize a whole image, even if the object is not centered, which is the case of the CIFAR-10 dataset.

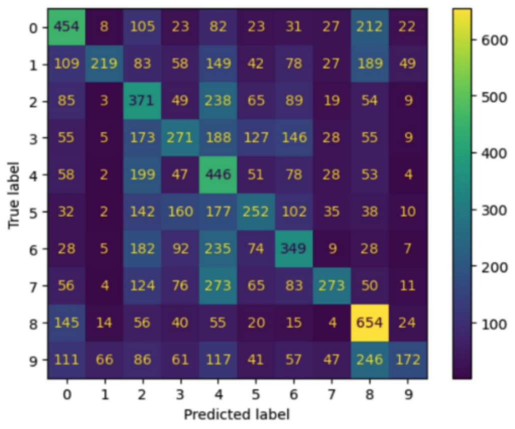
Contributions

While everyone collaboratively worked together on this project, Melissa did the most work in the experimental setup where she collected and manipulated code to insert data into Google Collab file and created the classifiers and neural networks, as well as observe what objects the classifiers were confused with the most. Ashish ran most of the codes on his local computer with the whole dataset to get the results faster than on Google Collab, and found a research paper to explore. Natalie found information on the classifiers and their code, and collected the data together to create conclusions about the classifiers.

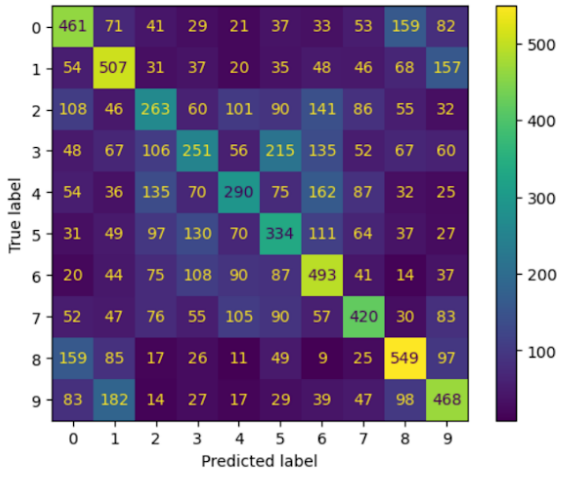
Appendix

Model type	Tuned hyperparameter	Test accuracy	Other Metric
kNN	k = 1	0.3461	See appendix figure 1
kNN	k = 5	0.3319	
kNN	k = 10	0.3325	
kNN	k = 50	0.3055	
kNN	k = 100	0.2945	
Logistic	C = 0	0.3849	
Logistic	C = 0.1	0.4027	See appendix figure 2
Logistic	C = 1	0.3956	
Logistic	C = 10	0.3846	
Logistic	C = 50	0.3850	
Feedforward Neural Network	Layerset = (256,128, 64)	0.1778	
Feedforward Neural Network	Layerset = (64,)	0.4037	
Feedforward Neural Network	Layerset = (128, 64)	0.4387	See appendix figure 4
Feedforward Neural Network	Layerset = (64, 32, 16)	0.4352	
Convolutional Neural Network	d = None	0.7310	
Convolutional Neural Network	d = 0.20	0.8315	
Convolutional Neural Network	d = 0.25	0.8346	See appendix figure 5
Convolutional Neural Network	d = 0.30	0.8328	

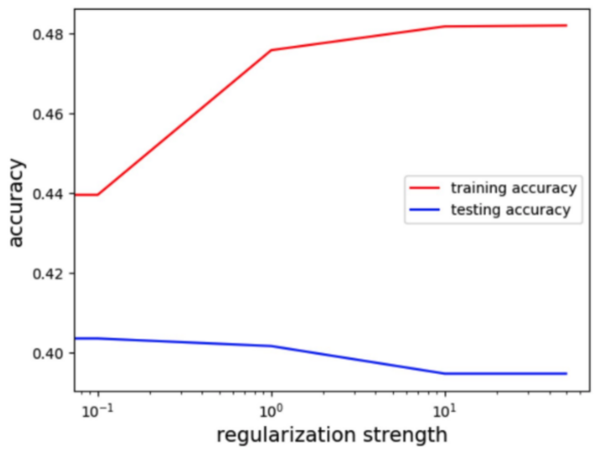
Appendix Table 1. Preliminary Results of All Models on Validation Dataset



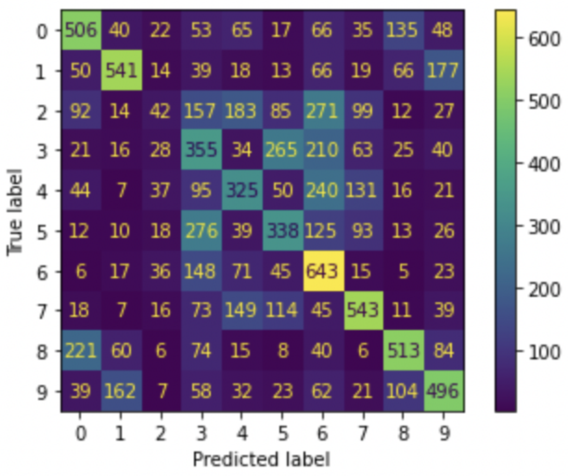
Appendix figure 1. Confusion Matrix for kNN Model (k = 1) Predictions on Validation Dataset



Appendix Figure 2. Confusion Matrix for Logistic Classifier (C = 0.1) Predictions on Validation Dataset

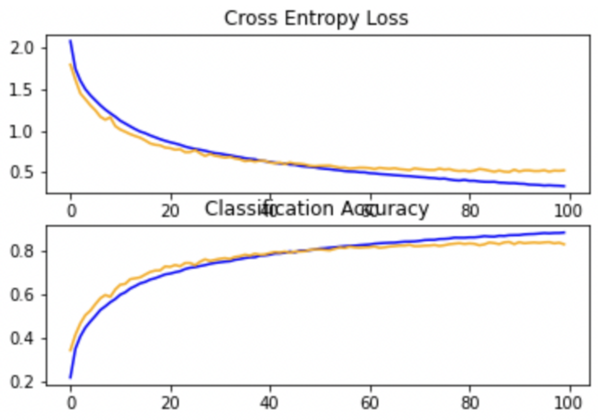


Appendix Figure 3. Accuracy as a Function of Regularization Strength For Logistic Classifiers

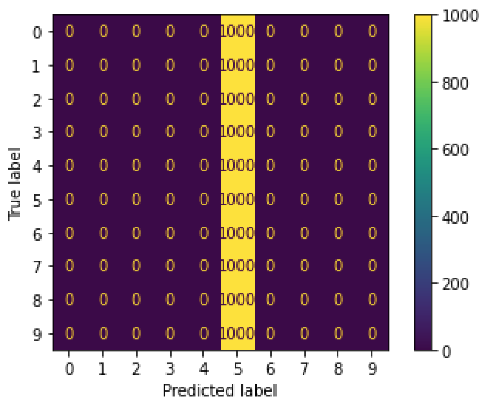


True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	506	40	22	53	65	17	66	35	135	48
1	50	541	14	39	18	13	66	19	66	177
2	92	14	42	157	183	85	271	99	12	27
3	21	16	28	355	34	265	210	63	25	40
4	44	7	37	95	325	50	240	131	16	21
5	12	10	18	276	39	338	125	93	13	26
6	6	17	36	148	71	45	643	15	5	23
7	18	7	16	73	149	114	45	543	11	39
8	221	60	6	74	15	8	40	6	513	84
9	39	162	7	58	32	23	62	21	104	496

Appendix Figure 4. Confusion Matrix for Feedback Neural Network (hidden_layer_sizes = (128,64)) Predictions on Validation Dataset

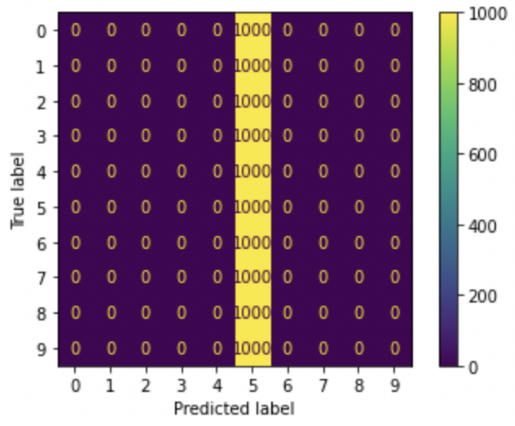


Appendix Figure 5. Cross Entropy Loss and Classification Accuracy Over the Number of Epochs for Convolutional Neural Network Predictions on Validation Dataset

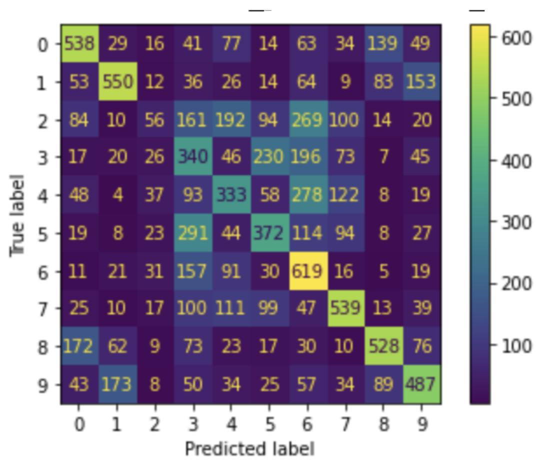


True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1000	0	0	0	0
1	0	0	0	0	0	1000	0	0	0	0
2	0	0	0	0	0	1000	0	0	0	0
3	0	0	0	0	0	1000	0	0	0	0
4	0	0	0	0	0	1000	0	0	0	0
5	0	0	0	0	0	1000	0	0	0	0
6	0	0	0	0	0	1000	0	0	0	0
7	0	0	0	0	0	1000	0	0	0	0
8	0	0	0	0	0	1000	0	0	0	0
9	0	0	0	0	0	1000	0	0	0	0

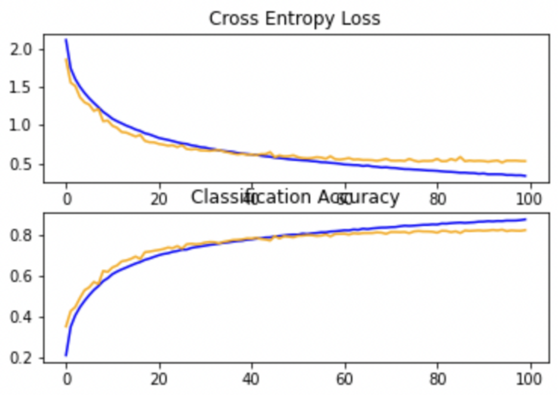
Appendix Figure 6. Confusion Matrix for kNN Model (k = 1) Predictions on Final Test Dataset



Appendix Figure 7. Confusion Matrix for Logistic Regression Model (C = 0.1) Predictions on Final Test Dataset



Appendix Figure 8. Confusion Matrix for Feedback Neural Network (hidden_layer_sizes = (128,64)) Predictions on Final Test Dataset



Appendix Figure 9. Cross Entropy Loss and Classification Accuracy Over the Number of Epochs for Convolutional Neural Network Predictions on Final Dataset