MSDS 458: Artificial Intelligence and Deep Learning

Assignment 2

October 15, 2025

Comparison between regular deep neural network and convolutional neural networks on CIFAR-10 dataset

Yingyu Mao

Abstract

In this study, I compared the performance, training characteristics, and computational overhead of various deep neural network architectures, specifically Fully Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs), on the CIFAR-10 image classification dataset. The goal is to inform the selection of an appropriate deep learning model for an online marketplace interested in automated, image-based categorization. FCNNs proved fast to train but consistently showed low test accuracy, plateauing around 52-56%. CNNs, even with only two convolutional layers, delivered significantly better performance, reaching 80% accuracy when explicit regularization was applied. A slightly deeper 3-Conv2D-layer CNN with regularization further improved results to 84% test accuracy. The deepest architecture tested, ResNet-18, although required extensive training, achieved the highest accuracy near 88%. These results confirmed the advantages of CNNs and their hierarchical feature learning ability for image tasks and provided a practical recommendation for a robust architecture for the initial prototyping phase.

Keywords: Convolutional Neural Network, ResNet, generalization, CIFAR-10, image classification

1.0 Introduction

The human visual system can effortlessly parse complex scenes, identifying objects, faces, and textures in a fraction of a second. Replicating this capability in machines has been a long-standing challenge in artificial intelligence. For decades, traditional computer vision techniques relied on hand-crafted features and algorithms, which often struggled with the immense variability of the real world. Deep learning, particularly Convolutional Neural Networks (CNNs), has induced a paradigm shift in computer vision. Inspired by the biological processes of the animal visual cortex, CNNs use similar receptive fields that make them uniquely adept at processing data with a grid-like structure, such as images. By learning hierarchical representations of features, from simple edges and corners to complex object parts, CNNs have largely enhanced performance benchmarks across a multitude of domains and fueled many advances in computer vision, such as real-time object detection systems in autonomous vehicles, pixel-accurate semantic segmentation for medical image analysis, facial recognition in biometric security systems. Furthermore, their principles have been extended to tasks beyond static images, revolutionizing video analysis (3D CNN, (Ji, et al. 2013)), and other applications like image generation (GAN (Goodfellow, et al. 2014) and VAE (Kingma and Welling 2013)).

At a hypothetical company that runs an online marketplace for used goods, the management is interested in implementing modern CNN technologies to perform automated image-based categorization. They want to determine the appropriate neural network architecture, their training difficulties, and computation overhead, before investing in collecting and labeling hundreds of thousands of their own user-uploaded images (which are messy, cluttered, and poorly lit).

CIFAR-10 is a well-established, publicly available dataset of 60,000 small color images across 10 classes. Adopting it for the initial search of neural network architecture and hyperparameters standardizes the evaluation process, accelerates model prototyping and research, reduces computational costs, and provides a reliable indicator of a model's potential performance on the more complex business-specific tasks. In this study, I will compare the classification performance of different deep learning architectures.

2.0 Literature review

The foundational architecture for modern CNNs was established by LeNet-5 (LeCun, et al. 1998), which successfully demonstrated the application of convolutional layers, pooling, and fully connected layers for digit recognition. The modern era of CNN evolution was catalyzed by AlexNet (Krizhevsky, Sutskever and Hinton 2012), which revealed that a deep CNN could dramatically outperform all existing methods on the large-scale ImageNet dataset, by leveraging the use of ReLU activation functions, batch normalization, and efficient training on GPUs. This was followed by architectures like VGGNet (Simonyan and Zisserman 2014), which showed that the depth of the network (16-19 weight layers) is a critical component for high performance. Then GoogleLeNet (Szegedy, et al. 2014) further advanced large image dataset prediction performance by using inception modules that capture patterns at different scales. Later, He et al. (He, et al. 2015) introduced the "skip connection" or "residual block", which allowed gradients to flow efficiently through a much deeper network, enabling the training of networks with hundreds or even thousands of layers (e.g., ResNet-152). This topology significantly improved accuracy and became the default backbone for countless vision tasks. These architectures generated image classification models that can be used in transfer learning to perform different end tasks. They made the foundation for different variations of CNN classifiers that incorporate special types of operations (such as the squeeze-and-excitation blocks (Hu, et al. 2017) and attention mechanism

(Wang, et al. 2017) that aimed at capturing channel-wise correlations), and other models developed for image segmentation, object detection, etc.

3.0 Methods

3.1 Data preprocessing

The CIFAR-10 dataset consists of 50K training images and 10K test images in 10 classes. Each image has 32-pixel width, 32-pixel height and 3 channels (RGB), with pixel values ranging from 0 to 255. The pixel values were first scaled by its range and then per-pixel mean subtracted. The resulting images have higher contrast than scaled-only images as shown in Figure 1. These images were used for training all models in this study. The 50K training set was also split into 0.9 training set and 0.1 validation set. Both the validation and test sets were scaled and transformed similarly with the per-pixel mean learned from the training set.

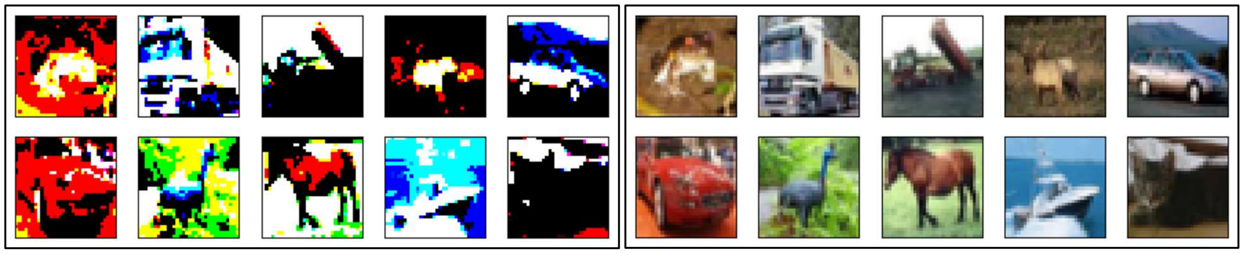


Figure 1 Images after per-pixel mean subtraction (left) have better contrasts than the original scaled images(right).

3.2 Experimental design

To compare the performance of FCNNs, the original architectural components of the LeNet-5, and a modern deep architecture, ResNet, the following experiments were performed:

Experiments	**Architecture**	**No. of parameters**
2-layer FCNN	Input: image tensors of shape [32,32,3] Flatten: data vector of shape [32*32*3,] Dense-1: 250 units Dense-2: 10 units	**Total params:** 770,760 (2.94 MB) **Trainable params:** 770,760 (2.94 MB) **Non-trainable params:** 0 (0.00 B)
2-layer FCNN with regularization	Input: image tensors of shape [32,32,3] Flatten: data vector of shape [32*32*3,] Dense-1: 500 units, L2 regularization Dense-2: 10 units	**Total params:** 770,760 (2.94 MB) **Trainable params:** 770,760 (2.94 MB) **Non-trainable params:** 0 (0.00 B)
3-layer FCNN	Input: image tensors of shape [32,32,3] Flatten: data vector of shape [32*32*3,] Dense-1: 300 units Dense-2: 450 units Dense-3: 10 units	**Total params:** 1,061,860 (4.05 MB) **Trainable params:** 1,061,860 (4.05 MB) **Non-trainable params:** 0 (0.00 B)
3-layer FCNN with regularization	Input: image tensors of shape [32,32,3] Flatten: data vector of shape [32*32*3,] Dense-1: 350 units, L2 regularization Dense-2: 150 units, L2 regularization	**Total params:** 1,129,710 (4.31 MB) **Trainable params:** 1,129,710 (4.31 MB) **Non-trainable params:** 0 (0.00 B)

	Dense-3: 10 units	
3-Conv2D-layer CNN	Input: image tensors of shape [32,32,3] Conv2D: 224 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Conv2D: 448 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Flatten Dense-1: 256 units BatchNormalization Dense-2: 10 units	**Total params:** 5,042,506 (19.24 MB) **Trainable params:** 5,041,994 (19.23 MB) **Non-trainable params:** 512 (2.00 KB)
2-Conv2D-layer CNN with regularization	Input: image tensors of shape [32,32,3] Conv2D: 256 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Dropout (0.4) Conv2D: 448 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Dropout (0.3) Flatten Dense-1: 384 units, L2 regularization BatchNormalization Dropout (0.4) Dense-2: 10 units	**Total params:** 6,206,474 (23.68 MB) **Trainable params:** 6,205,706 (23.67 MB) **Non-trainable params:** 768 (3.00 KB)
3-Conv2D-layer CNN	Input: image tensors of shape [32,32,3] Conv2D: 120 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Conv2D: 512 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Conv2D: 768 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Flatten Dense-1: 256 units BatchNormalization Dense-2: 10 units	**Total params:** 4,886,826 (18.64 MB) **Trainable params:** 4,886,314 (18.64 MB) **Non-trainable params:** 512 (2.00 KB)
3-Conv2D-layer CNN with regularization	Input: image tensors of shape [32,32,3] Conv2D: 256 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Dropout (0.4) Conv2D: 448 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Dropout (0.3) Conv2D: 1024 filters, 3*3 kernel size, stride 1 MaxPool2D: 2*2 kernel size, stride 2 Dropout (0.4) Flatten Dense-1: 384 units, L2 regularization BatchNormalization Dropout (0.3) Dense-2: 10 units	**Total params:** 6,748,234 (25.74 MB) **Trainable params:** 6,747,466 (25.74 MB) **Non-trainable params:** 768 (3.00 KB)
Resnet-18	Input: image tensors of shape [32,32,3]	**Total params:** 4,334,026 (16.53 MB)

	Conv2D: 64 filters, 3*3 kernel size, stride 1, same padding, no bias BatchNormalization ReLU layer Residual block: 64 filters (3x) Residual block: 128 filters (3x) Residual block: 256 filters (3x) GlobalAveragePooling2D Dense: 10 units Residual block: Conv2D -> BatchNormalization -> ReLU -> Conv2D -> BatchNormalization -> identity shortcut -> ReLU	**Trainable params:** 4,327,754 (16.51 MB) **Non-trainable params:** 6,272 (24.50 KB)
Test effect of learning rate on training of deeper networks in comparison with shallower networks	Compare: 3-layer FCNN with regularization 3-CNN-layer with regularization Resnet-18	

All hidden layers used ReLU for activation and He normal distribution rules for weight initiation (He, Zhang, et al. 2015). The output dense layer used Softmax for activation. The number of units or the number of filters, and the regularization strength for all architectures, except Resnet-18, were determined using `keras tuner` hyperband search (Li, et al. 2016). The settings of Resnet mostly replicates those used in the original paper (He, et al. 2015).

I compared the training time (on a local 2442 MB GPU for CNN models) and per class prediction performance of different models.

4.0 Model Training Results

All models were trained using Nadam optimizer, a variant of Adam in combination with the Nesterov trick. The loss function used was categorical cross entropy.

The learning process was controlled by a learning rate scheduler:

```
tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=1e-3,
decay_steps=500, decay_rate=0.95)
```

Early stopping was also used to prevent overfitting:

```
tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=3)
```

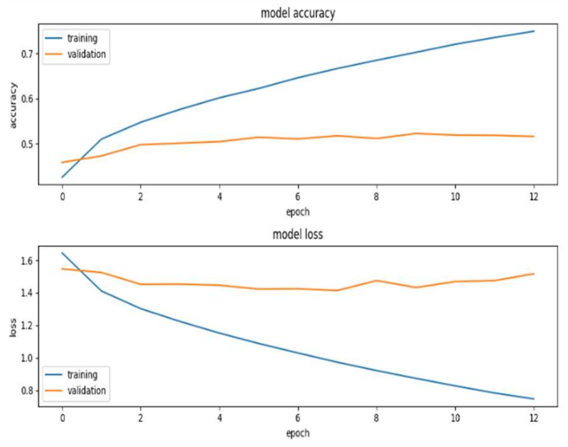
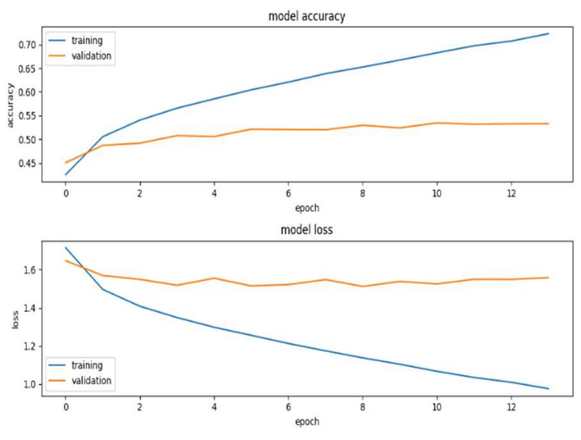
As a comparison, the original optimizer from the ResNet paper was also used to train ResNet:

```
tf.keras.optimizers.SGD(momentum=0.9, weight_decay=0.0001,
learning_rate=0.1)
```

4.1 Experiment 1 and 2: 2-layer FCNN with/without regularization

Regular FCNN were fast to train. Their prediction on the test set plateaued at about 0.5 accuracy (Table 1). Further training seemed to result in overfitting the training set, indicating the limitation of FCNN architecture in capturing patterns that generalize well. L2 regularization only managed overfitting to some degree and marginally increased test performance. This observation agrees with Zhang et al.'s notion that neural networks rely mostly on implicit regularization (the stochastic nature of the training process) to control generalization error, rather than explicit regularizations, such as weight decay or dropouts (Zhang, et al. 2016).

Table 1 Training performance of 2-layer FCNNs

Model	Training time	Training history	Performance on test set
2-layer FCNN	28.6 s		
2-layer FCNN with regularization	31.6 s		

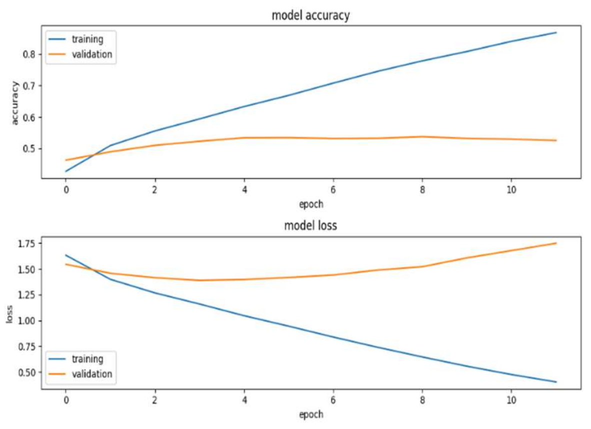
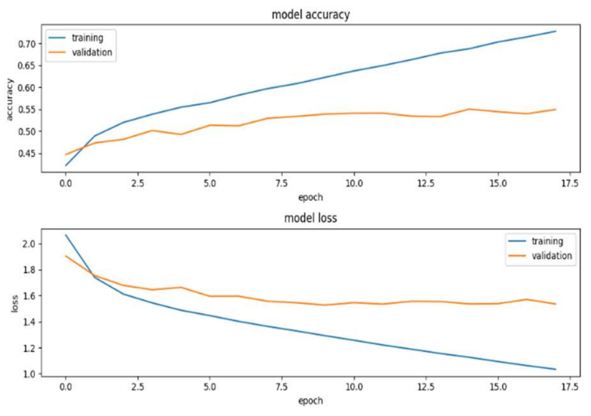
```
Classification Report
                 precision    recall  f1-score   support

           0         0.57      0.55      0.56      1000
           1         0.65      0.60      0.63      1000
           2         0.42      0.39      0.40      1000
           3         0.33      0.33      0.33      1000
           4         0.43      0.53      0.47      1000
           5         0.43      0.40      0.41      1000
           6         0.59      0.51      0.55      1000
           7         0.58      0.61      0.59      1000
           8         0.66      0.65      0.66      1000
           9         0.54      0.61      0.57      1000

    accuracy                             0.52     10000
   macro avg         0.52      0.52      0.52     10000
weighted avg         0.52      0.52      0.52     10000

Accuracy Score: 0.5178
Root Mean Square Error: 3.0178303464575342
```

```
Classification Report
                 precision    recall  f1-score   support

           0         0.59      0.66      0.62      1000
           1         0.68      0.61      0.64      1000
           2         0.49      0.37      0.42      1000
           3         0.37      0.37      0.37      1000
           4         0.48      0.44      0.46      1000
           5         0.47      0.40      0.43      1000
           6         0.48      0.73      0.58      1000
           7         0.63      0.58      0.60      1000
           8         0.70      0.64      0.67      1000
           9         0.57      0.62      0.59      1000

    accuracy                             0.54     10000
   macro avg         0.55      0.54      0.54     10000
weighted avg         0.55      0.54      0.54     10000

Accuracy Score: 0.5425
Root Mean Square Error: 2.943874997346185
```

4.2 Experiment 3 and 4: 3-layer FCNN with/without regularization

The 3-layer FCNNs were also fast to train. It only slightly increased the test performance compared to the 2-layer FCNN (Table 2). Similar saturation of test performance was also observed. L2 regularization had some control of the generalization gap but only marginally increased test performance.

Table 2 Training performance of 3-layer FCNNs

Model	Training time	Training history	Performance on test set
3-layer FCNN	32.9 s		Classification Report precision recall f1-score support 0 0.58 0.63 0.61 1000 1 0.67 0.65 0.66 1000 2 0.51 0.35 0.41 1000 3 0.35 0.40 0.38 1000 4 0.40 0.52 0.45 1000 5 0.45 0.42 0.44 1000 6 0.62 0.52 0.57 1000 7 0.58 0.60 0.59 1000 8 0.68 0.70 0.69 1000 9 0.60 0.60 0.60 1000 accuracy 0.54 10000 macro avg 0.55 0.54 0.54 10000 weighted avg 0.55 0.54 0.54 10000 Accuracy Score: 0.5394 Root Mean Square Error: 2.8864337858333076
3-layer FCNN with regularization	42.6 s		Classification Report precision recall f1-score support 0 0.61 0.59 0.60 1000 1 0.71 0.63 0.67 1000 2 0.46 0.44 0.45 1000 3 0.39 0.39 0.39 1000 4 0.47 0.51 0.49 1000 5 0.50 0.40 0.45 1000 6 0.58 0.64 0.61 1000 7 0.61 0.63 0.62 1000 8 0.61 0.74 0.67 1000 9 0.61 0.60 0.60 1000 accuracy 0.56 10000 macro avg 0.56 0.56 0.55 10000 weighted avg 0.56 0.56 0.55 10000 Accuracy Score: 0.5565 Root Mean Square Error: 2.9068023668629417

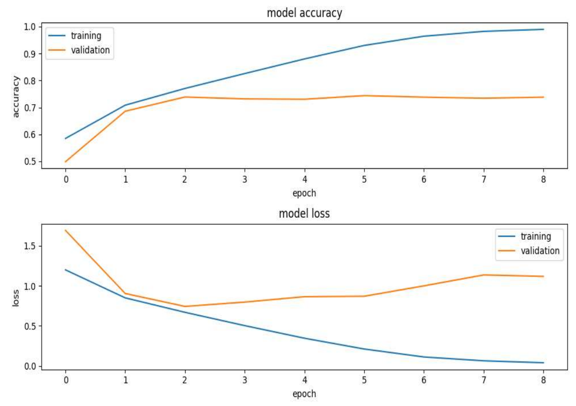
4.3 Experiment 5 and 6: 2-Conv2D-layer CNN with/without regularization

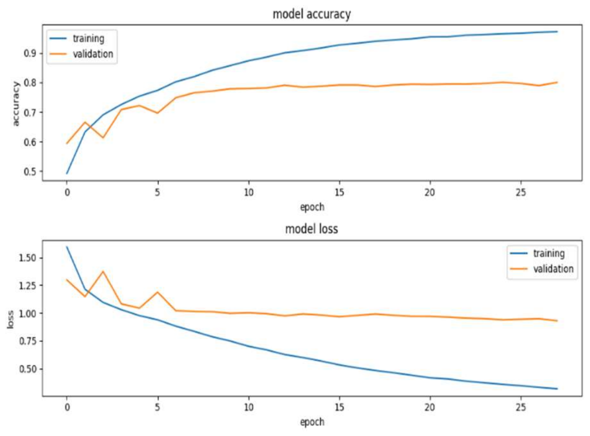
With only 2 Convolutional and 2 maximal pooling layers, the CNN models' test performance improves significantly (Table 3), although these models required a longer time to train than the FCNN. The computation complexity comes from the data structure handled by the processor during training: FCNN has 2D matrices (batch, flattened images) flowing through the forward pass, while CNN network processes 4D tensors (batch, image width, image height, number of channels/feature maps).

The activation values of feature maps at different layers of a CNN can be visualized with heatmaps (data in python notebook). Interestingly, the feature maps in the early layers seemed to act as edge detectors that traced the shapes of an object in the original image. The deeper feature maps became more abstract and less visually interpretable, as these turned into encodings of the abstract concept of a frog or a dog. The sparsity of the active filters also increased in later layers, which might be due to the use of ReLU activation and loss of irrelevant features. Using Dropout layers between these convolutional layers was an effective regularization. The regularized network took significantly longer (almost 10 times total wall time and about 5 times more per epoch) to reach the early stopping criteria. Extra dropout layers did add more computation time. Additionally, solutions found with regularization had a steady increase in generalization accuracy (validation accuracy).

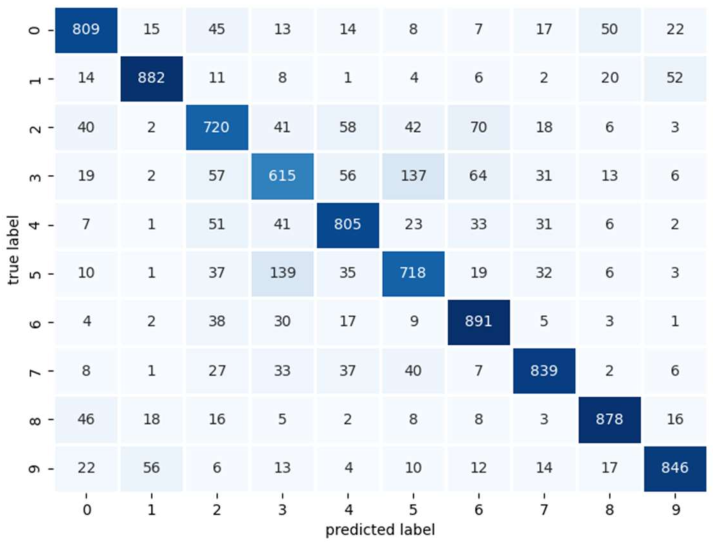
Table 3 Performance of 2-Conv2D-layer CNNs

Model	Training time	Training history	Performance on test set

2-Conv2D-layer	3min 3s		`Classification Report`
			`              precision    recall  f1-score   support`
			`           0       0.69      0.81      0.74      1000`
			`           1       0.90      0.79      0.84      1000`
			`           2       0.72      0.59      0.65      1000`
			`           3       0.57      0.57      0.57      1000`
			`           4       0.62      0.82      0.71      1000`
			`           5       0.67      0.66      0.66      1000`
			`           6       0.89      0.69      0.78      1000`
			`           7       0.80      0.76      0.78      1000`
			`           8       0.78      0.86      0.82      1000`
			`           9       0.82      0.81      0.82      1000`
			`    accuracy                           0.74     10000`
			`   macro avg       0.75      0.74      0.74     10000`
			`weighted avg       0.75      0.74      0.74     10000`
			`Accuracy Score: 0.7365`
			`Root Mean Square Error: 2.125864530020669`

2-Conv2D-layer with regularization	23min 59s		`Classification Report`
			`              precision    recall  f1-score   support`
			`           0       0.83      0.81      0.82      1000`
			`           1       0.90      0.88      0.89      1000`
			`           2       0.71      0.72      0.72      1000`
			`           3       0.66      0.61      0.63      1000`
			`           4       0.78      0.81      0.79      1000`
			`           5       0.72      0.72      0.72      1000`
			`           6       0.80      0.89      0.84      1000`
			`           7       0.85      0.84      0.84      1000`
			`           8       0.88      0.88      0.88      1000`
			`           9       0.88      0.85      0.86      1000`
			`    accuracy                           0.80     10000`
			`   macro avg       0.80      0.80      0.80     10000`
			`weighted avg       0.80      0.80      0.80     10000`
			`Accuracy Score: 0.8003`
			`Root Mean Square Error: 1.8338484124921557`

The performance differences among classes indicate that categories 2-5 ('bird', 'cat', 'deer', 'dog') had more misclassifications. The confusion matrix of the regularized CNN (Figure 2 top) reveals that some ‘bird’ images were classified as ‘airplane’ or 'cat', 'deer' or 'dog' and there were not enough features in the images to distinguish 'cat', 'deer' and 'dog', especially between 'cat' and 'dog', with the highest number of cross misclassifications. t-SNE projections of the activations of the test set extracted from the last dense layer before the output layer (Figure 2 bottom) also show similar findings: strong overlap of the ‘cat’ and ‘dog’ clusters. ‘bird’ data points appear more scattered and overlap with multiple classes. ‘truck’, ‘automobile’, ‘ship’ and ‘frog’ clusters appear more defined.



	0	1	2	3	4	5	6	7	8	9
0	809	15	45	13	14	8	7	17	50	22
1	14	882	11	8	1	4	6	2	20	52
2	40	2	720	41	58	42	70	18	6	3
3	19	2	57	615	56	137	64	31	13	6
4	7	1	51	41	805	23	33	31	6	2
5	10	1	37	139	35	718	19	32	6	3
6	4	2	38	30	17	9	891	5	3	1
7	8	1	27	33	37	40	7	839	2	6
8	46	18	16	5	2	8	8	3	878	16
9	22	56	6	13	4	10	12	14	17	846

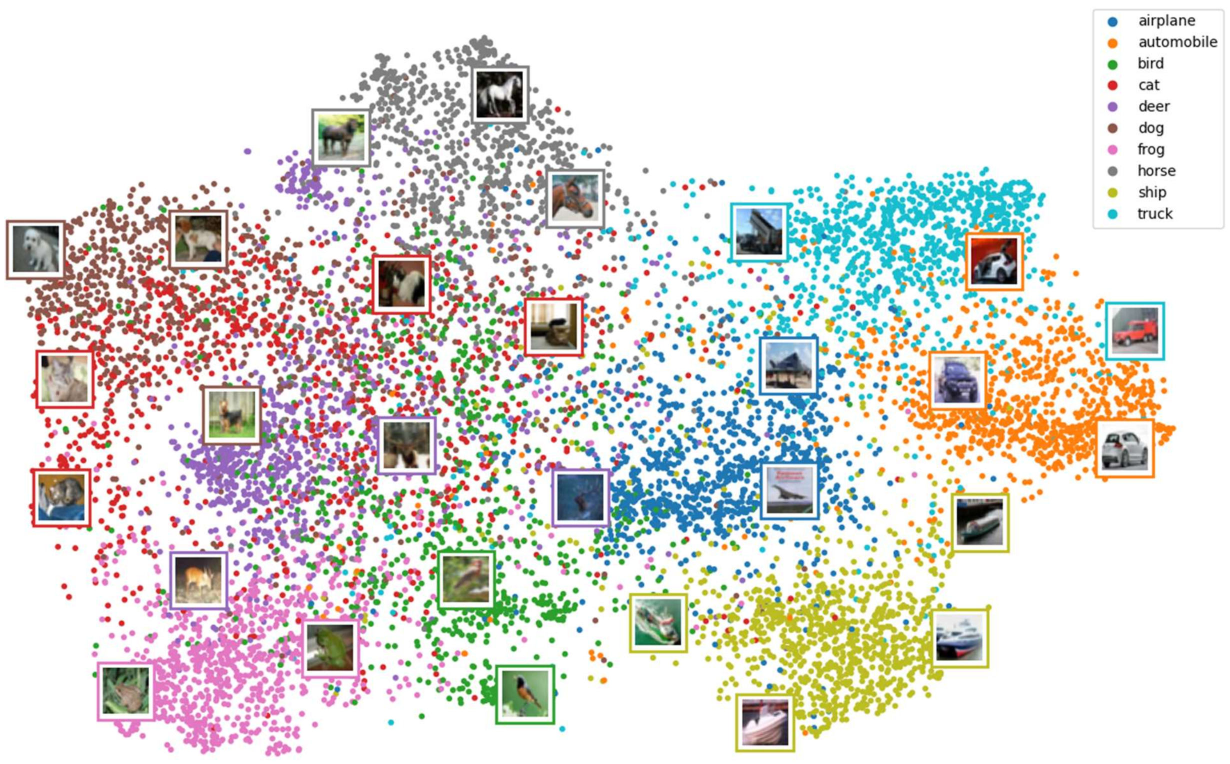
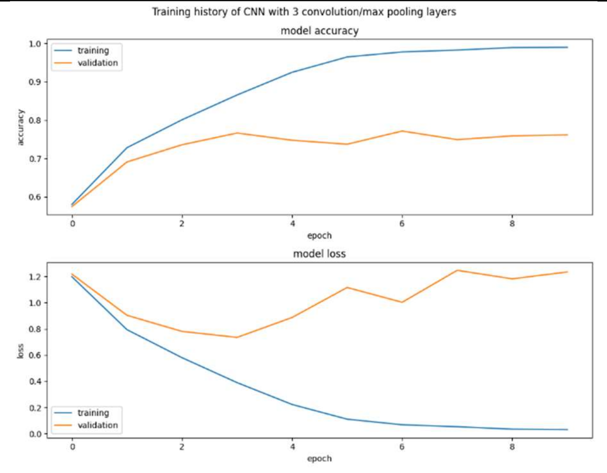
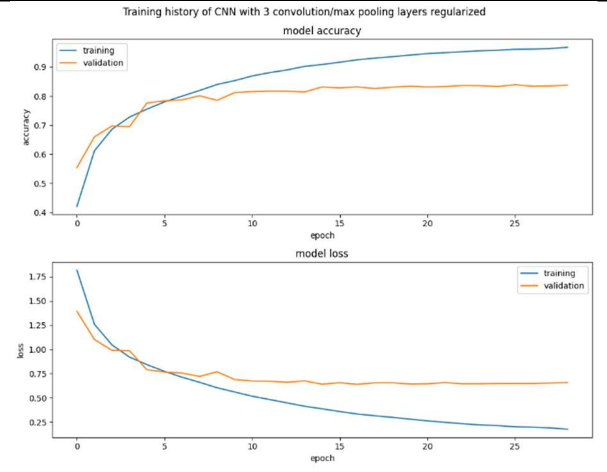


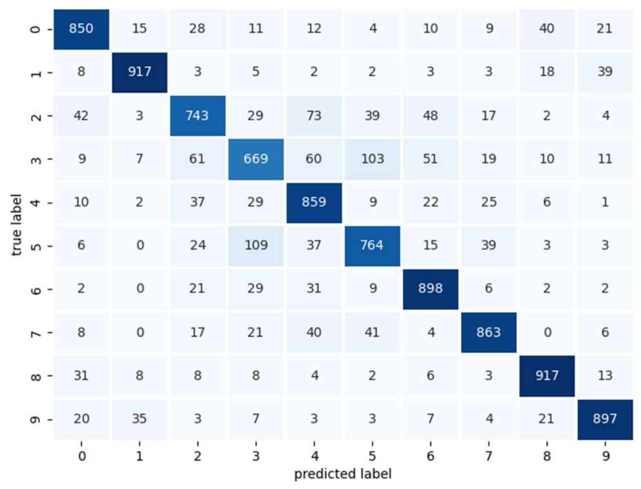
Figure 2 ***Top****: Confusion matrix of the test predictions by 2-Conv2D-layer regularized CNN.* ***Bottom****: t-SNE projection of the activations of the test set extracted from the last dense layer before the output layer.*

4.3 Experiment 7 and 8: 3-Conv2D-layer CNN with/without regularization

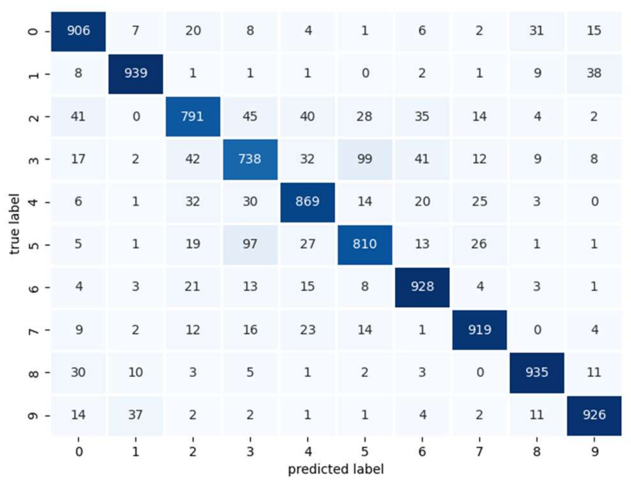
The 3-Conv2D-layer models further increased the test performance of the CNN model (Table 4). More convolutional layers correspond to further abstraction of the image data. Multiple layers and types of explicit regularizations reduced the generalization gap and improved prediction performance on the difficult categories mentioned above (Figure 3). The t-SNE projection of the activations of the test set from the last dense layer before output also reveals slight improvement in separation of 'cat' and 'dog' data points. The 'deer' cluster is positioned away from the 'cat' and 'dog' cluster and close to the 'horse' cluster. The other clusters also became tighter and have clearer boundaries.

Table 4 With only 2 Convolutional and 2 Performance of 3-Conv2D-layer CNNs

Model	Training time	Training history	Performance on test set
3-Conv2D -layer	5min 1s		Classification Report precision recall f1-score support 0 0.79 0.80 0.79 1000 1 0.93 0.83 0.88 1000 2 0.64 0.74 0.69 1000 3 0.73 0.47 0.57 1000 4 0.74 0.70 0.72 1000 5 0.75 0.66 0.70 1000 6 0.73 0.88 0.80 1000 7 0.72 0.88 0.79 1000 8 0.83 0.89 0.86 1000 9 0.87 0.83 0.85 1000 accuracy 0.77 10000 macro avg 0.77 0.77 0.77 10000 weighted avg 0.77 0.77 0.77 10000 Accuracy Score: 0.7688 Root Mean Square Error: 1.941854783448031
3-Conv2D -layer with regularization	57min 2s		Classification Report precision recall f1-score support 0 0.86 0.85 0.86 1000 1 0.93 0.92 0.92 1000 2 0.79 0.74 0.76 1000 3 0.73 0.67 0.70 1000 4 0.77 0.86 0.81 1000 5 0.78 0.76 0.77 1000 6 0.84 0.90 0.87 1000 7 0.87 0.86 0.87 1000 8 0.90 0.92 0.91 1000 9 0.90 0.90 0.90 1000 accuracy 0.84 10000 macro avg 0.84 0.84 0.84 10000 weighted avg 0.84 0.84 0.84 10000 Accuracy Score: 0.8377 Root Mean Square Error: 1.5958383376770968



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	850	15	28	11	12	4	10	9	40	21
1	8	917	3	5	2	2	3	3	18	39
2	42	3	743	29	73	39	48	17	2	4
3	9	7	61	669	60	103	51	19	10	11
4	10	2	37	29	859	9	22	25	6	1
5	6	0	24	109	37	764	15	39	3	3
6	2	0	21	29	31	9	898	6	2	2
7	8	0	17	21	40	41	4	863	0	6
8	31	8	8	8	4	2	6	3	917	13
9	20	35	3	7	3	3	7	4	21	897



true label \ predicted label	0	1	2	3	4	5	6	7	8	9
0	906	7	20	8	4	1	6	2	31	15
1	8	939	1	1	1	0	2	1	9	38
2	41	0	791	45	40	28	35	14	4	2
3	17	2	42	738	32	99	41	12	9	8
4	6	1	32	30	869	14	20	25	3	0
5	5	1	19	97	27	810	13	26	1	1
6	4	3	21	13	15	8	928	4	3	1
7	9	2	12	16	23	14	1	919	0	4
8	30	10	3	5	1	2	3	0	935	11
9	14	37	2	2	1	1	4	2	11	926

*Figure 3 **Left**: Confusion matrix of test predictions by 3-Conv2D-layer regularized CNN. **Right**: Confusion matrix of the test set of ResNet-18.*

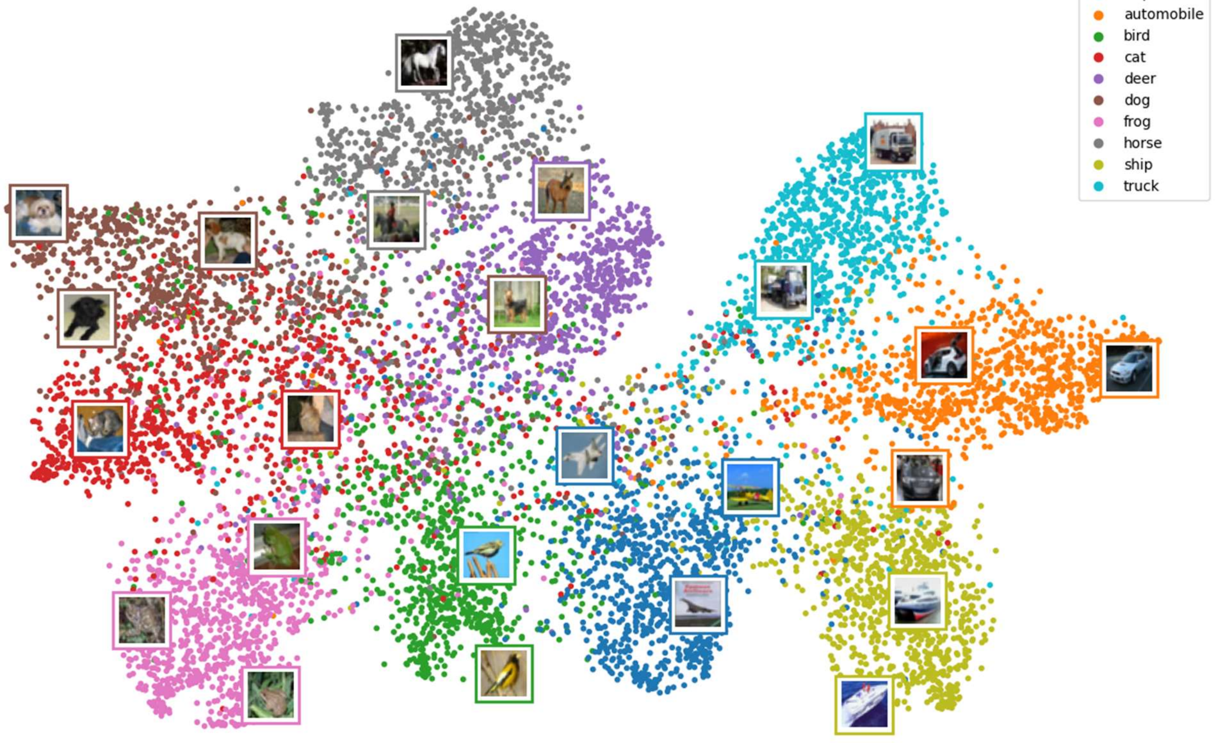
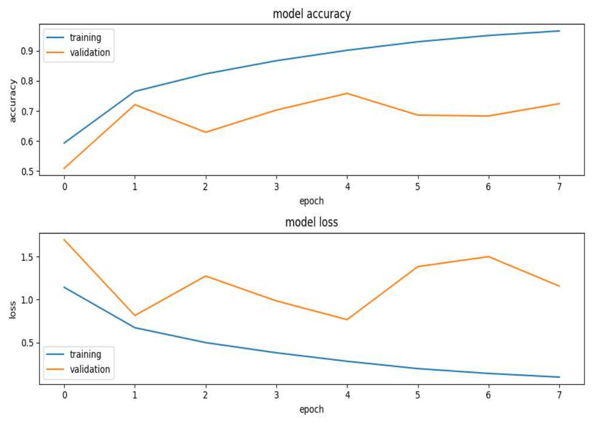
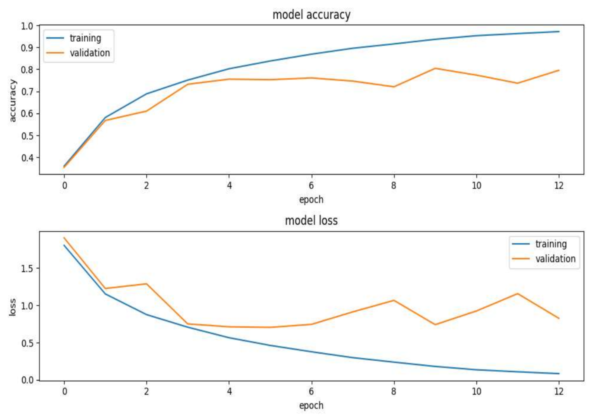


Figure 4 t-SNE projection of the activations of the test set extracted from the last dense layer before the output layer in the 3-Conv2D-layer regularized CNN model.

4.4 Experiment 9: ResNet-18

ResNet architecture consists of a stack of convolutional layers (no max pooling in between but a final global pooling) and relies mostly on batch normalization and residual shortcuts to resolve vanishing and exploding gradient problems in deep networks. This deep CNN took more than twice the time per echo to train than the regularized 3-Conv2D-layer model. However, neither Nadam optimizer at 0.001 learning rate nor plain SGD optimizer at 0.1 learning rate regularized with early stopping resulted in a model with better test performance compared to the regularized 3-Conv2D-layer model, even though they reached similar training accuracy (Table 5).

Table 5 Performance of ResNet-18

Model	Training time	Training history	Performance on test set
ResNet-18 trained with Nadam with the learning rate scheduler	19min 46s		Classification Report precision recall f1-score support 0 0.87 0.70 0.78 1000 1 0.51 0.98 0.67 1000 2 0.67 0.78 0.72 1000 3 0.74 0.50 0.60 1000 4 0.84 0.70 0.76 1000 5 0.74 0.68 0.71 1000 6 0.95 0.59 0.73 1000 7 0.95 0.56 0.70 1000 8 0.89 0.85 0.87 1000 9 0.58 0.88 0.70 1000 accuracy 0.72 10000 macro avg 0.77 0.72 0.72 10000 weighted avg 0.77 0.72 0.72 10000 Accuracy Score: 0.7233 Root Mean Square Error: 2.330986915450192
ResNet-18 trained with SGD with 0.1 learning rate	38min 17s		Classification Report precision recall f1-score support 0 0.90 0.77 0.83 1000 1 0.98 0.78 0.87 1000 2 0.73 0.76 0.74 1000 3 0.60 0.73 0.66 1000 4 0.81 0.80 0.80 1000 5 0.63 0.81 0.71 1000 6 0.97 0.63 0.77 1000 7 0.83 0.87 0.85 1000 8 0.90 0.92 0.91 1000 9 0.84 0.91 0.87 1000 accuracy 0.80 10000 macro avg 0.82 0.80 0.80 10000 weighted avg 0.82 0.80 0.80 10000 Accuracy Score: 0.7983 Root Mean Square Error: 1.7926237753639216

To reproduce the high performance reported in He et al.'s paper on ResNet, I performed Experiment 10.

4.5 Experiment 10: Higher learning rate for NNs of different depths

I compared the three different architectures and depths, FCNN (3 layers), LeNet-5-like CNN (3 Conv2D layers), and ResNet (18 Conv2D layers with batch normalization and skip connections), trained with the same optimizer with 0.1 learning rate and without early stopping, as the ResNet paper trained their model for 32K iterations at this learning rate before lowering it by 10 times for fine tuning. Using this training setup, the ResNet model's test accuracy increased to 0.88 with significant improvement on the difficult classes (Table 6, Figure 3 right). Similar t-SNE projection of the test activations extracted from the global average pooling layer before the output also shows better clustering of the 10 classes (Figure 5).

Interestingly, the training history of ResNet-18 showed some bumpy ups and downs of the validation accuracy up to about 50 epoch, but it then turned into a smooth and steady curve. This dynamic was not observed in the other two shallower networks. The learning rate was simply too high for FCNN, as the training accuracy plateaued at 0.6 and the learning curves were also bumpy. For the 3-Conv2D-layer regularized CNN, the training trajectory was smooth, but the validation curves did not converge to a smooth learning curve and did not further improve the test accuracy with the longer training.

Training history reveals how optimizers navigate the loss landscape. There was empirical evidence of a reproducible correlation between the flatness of the minima and the generalization performance (Jiang, et al. 2019). Flatter minima correspond to models that are more robust to minor perturbations in their parameter, that is, more relevant features that can generalize well. Numerous works studied the connection between the size/depth of the network and the distribution of the loss minima in the solution space. Baldassi et al. (2021) showed that high-margin (robust) solutions tend to cluster together. The coalescence of minima results in dense regions of solutions over long distances that are proportional to the size of the network. Verpoort et al. (2020) characterized local minima and transition state using computational energy landscapes techniques and found lower barriers between local minima in terms of both local and global organization in the loss landscape of a deeper network. These works can be used to explain how the deep architecture of ResNet, despite having less parameters than the 3-Conv2D CNN, had better learning outcome. The depth of ResNet allows its loss landscape to form larger regions of high-margin solution clusters. The higher learning rate not only helped with training earlier layers but also enabled navigation through the low-energy barriers between clusters of local minima. In combination with enough training epochs, the optimizer had higher probability of landing in the flatter regions of the loss landscape corresponding to robust solutions.

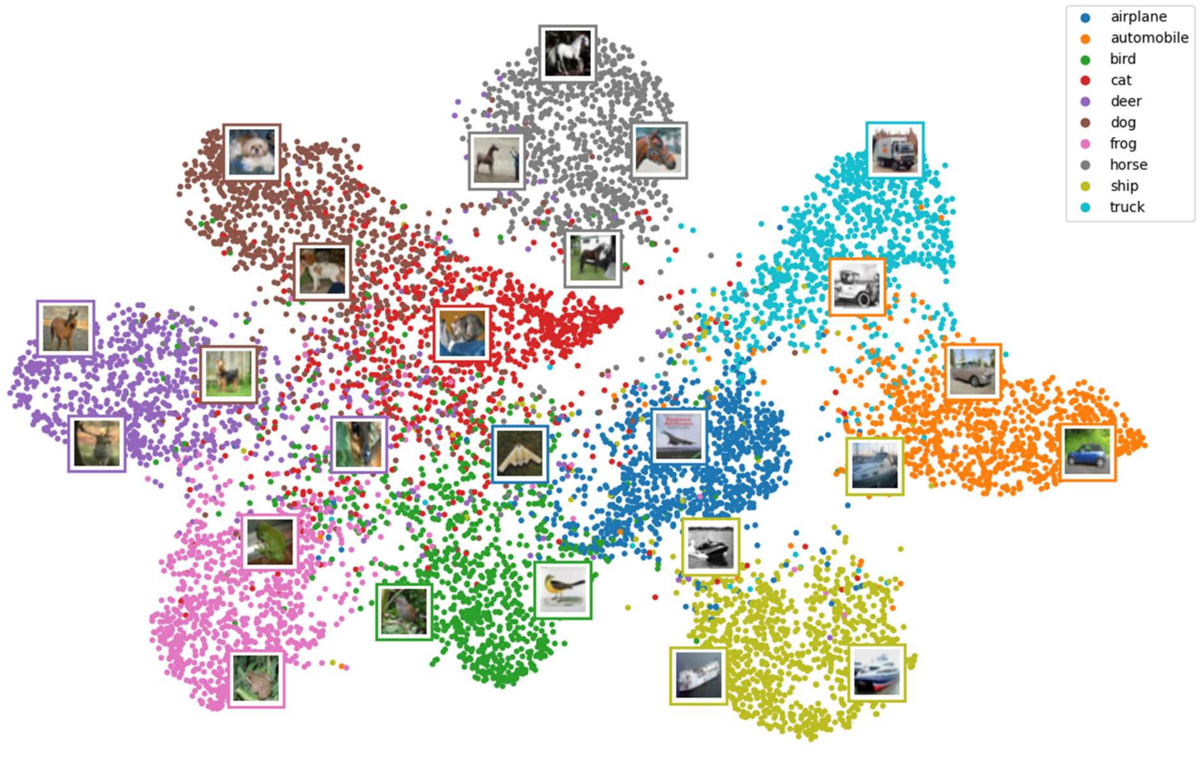
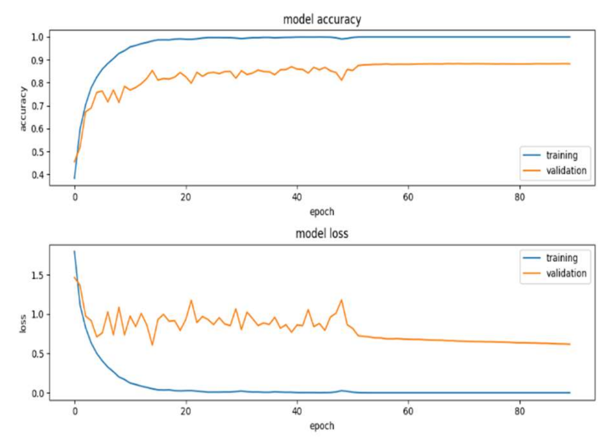
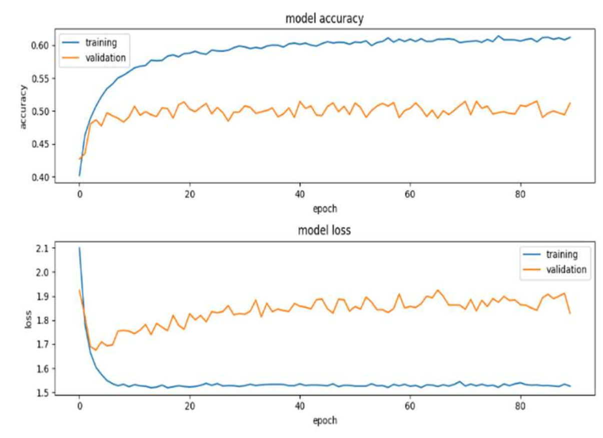
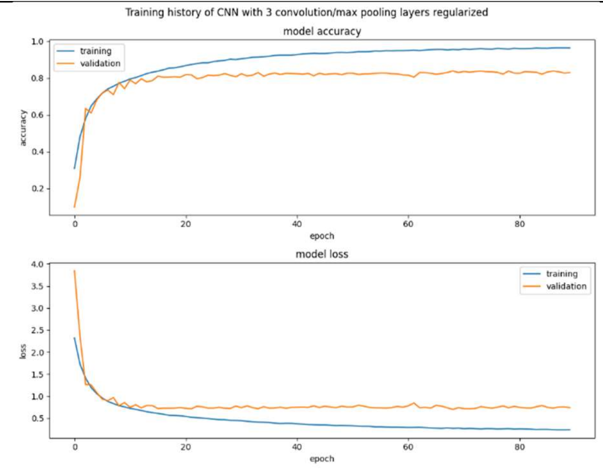


Figure 5 t-SNE projection of the activations of the test set extracted from the last dense layer before the output layer in ResNet-18 model.

Table 6 Performance of different models with a high learning rate

Model	Training time	Training history	Performance on test set
ResNet-18 trained with 0.1 learning rate and 90 epoch	5h 36min 26s		`Classification Report` `precision recall f1-score support` `0 0.87 0.91 0.89 1000` `1 0.94 0.94 0.94 1000` `2 0.84 0.79 0.81 1000` `3 0.77 0.74 0.75 1000` `4 0.86 0.87 0.86 1000` `5 0.83 0.81 0.82 1000` `6 0.88 0.93 0.90 1000` `7 0.91 0.92 0.92 1000` `8 0.93 0.94 0.93 1000` `9 0.92 0.93 0.92 1000` `accuracy 0.88 10000` `macro avg 0.88 0.88 0.88 10000` `weighted avg 0.88 0.88 0.88 10000` `Accuracy Score: 0.8761` `Root Mean Square Error: 1.4248157775656474`
3-layer FCNN trained with 0.1 learning rate and 90 epoch	1min 47s		`Classification Report` `precision recall f1-score support` `0 0.59 0.59 0.59 1000` `1 0.64 0.63 0.64 1000` `2 0.43 0.37 0.40 1000` `3 0.34 0.39 0.36 1000` `4 0.43 0.48 0.45 1000` `5 0.47 0.35 0.40 1000` `6 0.54 0.61 0.57 1000` `7 0.65 0.56 0.60 1000` `8 0.61 0.68 0.65 1000` `9 0.57 0.58 0.57 1000` `accuracy 0.52 10000` `macro avg 0.53 0.52 0.52 10000` `weighted avg 0.53 0.52 0.52 10000` `Accuracy Score: 0.5242` `Root Mean Square Error: 2.965231862772286`
3-Conv2D-layer CNN trained with 0.1 learning rate and 90 epoch	About 2h 30min		`Classification Report` `precision recall f1-score support` `0 0.85 0.85 0.85 1000` `1 0.95 0.89 0.92 1000` `2 0.78 0.75 0.76 1000` `3 0.73 0.68 0.70 1000` `4 0.75 0.87 0.81 1000` `5 0.79 0.77 0.78 1000` `6 0.84 0.91 0.87 1000` `7 0.89 0.83 0.86 1000` `8 0.89 0.91 0.90 1000` `9 0.89 0.90 0.89 1000` `accuracy 0.84 10000` `macro avg 0.84 0.84 0.84 10000` `weighted avg 0.84 0.84 0.84 10000` `Accuracy Score: 0.836` `Root Mean Square Error: 1.6406401189779556`

5.0 Conclusions

The summary of all models tested is shown below:

Models	**Size**	**Training time**	**Test accuracy**
2-layer FCNN	770,760 (2.94 MB)	28.6 s	0.52
2-layer FCNN with regularization	770,760 (2.94 MB)	31.6 s	0.54
3-layer FCNN	1,061,860 (4.05 MB)	32.9 s	0.54
3-layer FCNN with regularization	1,129,710 (4.31 MB)	42.6 s	0.56
3-Conv2D-layer CNN	5,042,506 (19.24 MB)	3min 3s	0.74
2-Conv2D-layer CNN with regularization	6,206,474 (23.68 MB)	23min 59s	0.80
3-Conv2D-layer CNN	4,886,826 (18.64 MB)	5min 1s	0.77
3-Conv2D-layer CNN with regularization	6,748,234 (25.74 MB)	57min 2s	0.84
ResNet-18	4,334,026 (16.53 MB)	5h 36min 26s	0.88

The performance winner is ResNet-18, achieving nearly 88% accuracy with less parameters. However, due to its deep architecture, it is more difficult and slower to train (on a low-end GPU on a personal computer). Due to shortness of computing resource, I did not perform data augmentation or continue onto the second phase of training for this model where the learning rate is reduced to 0.01 as described in the ResNet paper, which could potentially reach higher accuracy (to furth distinguish the 'cat' and 'dog' category).

With these observations from this preliminary test, I recommend the following to the management:

- Deeper CNN architectures outperform their shallower counterparts and can potentially reach higher performance through data augmentation and fine tuning.
- Ensure hardware setup (powerful GPUs) if it is necessary to train a deeper ResNet from scratch with their own image inventory.
- Consider transfer learning using various models (such as ResNet, Xception, EfficientNet, etc.) pre-trained on large image databases.

References

Baldassi, Carlo, Clarissa Lauditi, Enrico M. Malatesta, Gabriele Perugini, and Riccardo Zecchina. 2021. "Unveiling the structure of wide flat minima in neural networks." *arXiv* 2107.01163.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Networks." *arXiv* 1406.2661.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Deep Residual Learning for Image Recognition." *arXiv* 1512.03385.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." *arXiv* 1502.01852.

Hu, Jie, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. 2017. "Squeeze-and-Excitation Networks." *arXiv* 1709.01507.

Ji, S., W. Xu, M. Yang, and K. Yu. 2013. "3D Convolutional Neural Networks for Human Action Recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. 35, no. 1, pp. 221-231.

Jiang, Yiding, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2019. "Fantastic Generalization Measures and Where to Find Them." *arXiv* 1912.02178.

Kingma, Diederik P, and Max Welling. 2013. "Auto-Encoding Variational Bayes." *arXiv* 1312.6114.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks (AlexNet)." *NeurIPS Proceedings.*

LeCun, Yann, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE* vol. 86, no. 11, pp. 2278-2324.

Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization." *arXiv* 1603.06560.

Simonyan, Karen, and Andrew Zisserman. 2014. "Very Deep Convolutional Networks for Large-Scale Image Recognition (VGGNet)." *arXiv* 1409.1556.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. "Going Deeper with Convolutions." *arXiv* 1409.4842.

Verpoort, Philipp C., AlphaA. Lee, and David J. Wales. 2020. "Archetypal landscapes for deep neural networks." *PNAS* vol. 117 no. 36 21857-21864.

Wang, Fei, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. 2017. "Residual Attention Network for Image Classification." *arXiv* 1704.06904.

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. "Understanding deep learning requires rethinking generalization." *arXiv* 1611.03530.