EEE3314 Introduction to AI

Final Project Report

Team 9

Junsoo Ham	**Hyeonbeen Jeon**	**John Yechan Jo**	**Jin Chun**
2022144053	2020132002	2020142038	2021142198

December 8, 2025

Introduction

Throughout the semester in the "Introduction to AI" course, we covered a broad range of topics, starting with classical machine learning algorithms for supervised and unsupervised learning. In the second half of the course, we focused on deep learning and neural networks, including MLPs, CNNs, network generalization and optimization, transfer learning, and transformers, learning how to apply these methods to supervised learning problems.

In this final project, we solved various supervised and unsupervised learning problems—including curve fitting, dimension reduction of the CIFAR-10 dataset, face recognition with transfer learning, and training-free object localization—by leveraging deep learning frameworks and applying the various machine learning and deep learning methodologies learned throughout the semester.

Requirements

Table 1: Requirements

Package	**Question 1**	**Question 2**	**Question 3**	**Question 4**
python	3.12.12	3.12.12	3.12.12	3.12.12
numpy	2.0.2	2.0.2	2.3.3	2.3.3
torch	2.9.0+cu126	2.9.0+cu126	2.9.1+cu130	2.9.1+cu130
torchvision	–	0.24.0+cu126	0.24.1+cu130	0.24.1+cu130
matplotlib	3.10.0	3.10.0	3.10.6	3.10.6
opencv	–	–	–	4.12.0
pillow	–	–	–	11.3.0
IPython	8.12.3	8.12.3	–	–
sklearn	1.6.1	1.6.1	–	–

Table 1 summarizes the external libraries, frameworks, and their versions used to solve each question. Since the project was conducted collaboratively and each team member worked in a different environment, we explicitly listed the versions that were used. Although it is ideal to reproduce the environment using the exact versions specified, it should also be possible to build a compatible environment using alternative versions that maintain compatibility.

Question 1: Regression (Curve Fitting) Problem using MLPs

Data Preparation

- **Step 1: Generate Data.** We generate 262,144 (512×512) 2-dimension data points based on $f(x_1, x_2)$ where $-512 \leq x_1 \leq 512$ and $-512 \leq x_2 \leq 512$. Using `np.meshgrid`, these points are flattened to create input pairs, and $f(x_1, x_2)$ is calculated for each pair.
- **Step 2: Add Noise.** We add standard Gaussian noise: $\hat{f}(x_1, x_2) = f(x_1, x_2) + 0.1 \cdot \mathcal{N}(0, 1)$
- **Step 3: Standardize.** Z-score normalization is applied: $\frac{x-\mu}{\sigma}$
- **Step 4: Split Data.** The dataset is divided into training (70%), validation (15%), and test (15%) sets.

Hyperparameters (Common)

Category	Method	Value	Reason
Minibatch	-	256	Balance between convergence and overhead
Optimizer	Adam	lr=0.001	Adaptive learning rates for complex landscape
Initialization	He	-	Suitable for ReLU activation
Max Epoch	-	500	-

Q1-6: Single Hidden Layer MLP (Baseline)

(a) Nested CV Results

Table 2: 3-Fold CV Results for Single Hidden Layer MLP

# Hidden Neurons, H	10	100	200	500
Fold 1	$Err_{train} = 0.9715$, $Err_{val} = 0.9771$	$Err_{train} = 0.7175$, $Err_{val} = 0.7234$	$Err_{train} = 0.6744$, $Err_{val} = 0.6907$	$Err_{train} = 0.4551$, $Err_{val} = 0.4545$
Fold 2	$Err_{train} = 0.9777$, $Err_{val} = 0.9750$	$Err_{train} = 0.7250$, $Err_{val} = 0.7228$	$Err_{train} = 0.6740$, $Err_{val} = 0.6754$	$Err_{train} = 0.3267$, $Err_{val} = 0.3273$
Fold 3	$Err_{train} = 0.9702$, $Err_{val} = 0.9713$	$Err_{train} = 0.6713$, $Err_{val} = 0.6705$	$Err_{train} = 0.6137$, $Err_{val} = 0.6145$	$Err_{train} = 0.4348$, $Err_{val} = 0.4297$
Mean±std	0.9731 ± 0.0040 / 0.9745 ± 0.0029	0.7046 ± 0.0291 / 0.7056 ± 0.0304	0.6540 ± 0.0349 / 0.6602 ± 0.0403	$\mathbf{0.4055 \pm 0.0690}$ / $\mathbf{0.4038 \pm 0.0674}$

Best H = 500

(b) Visualization

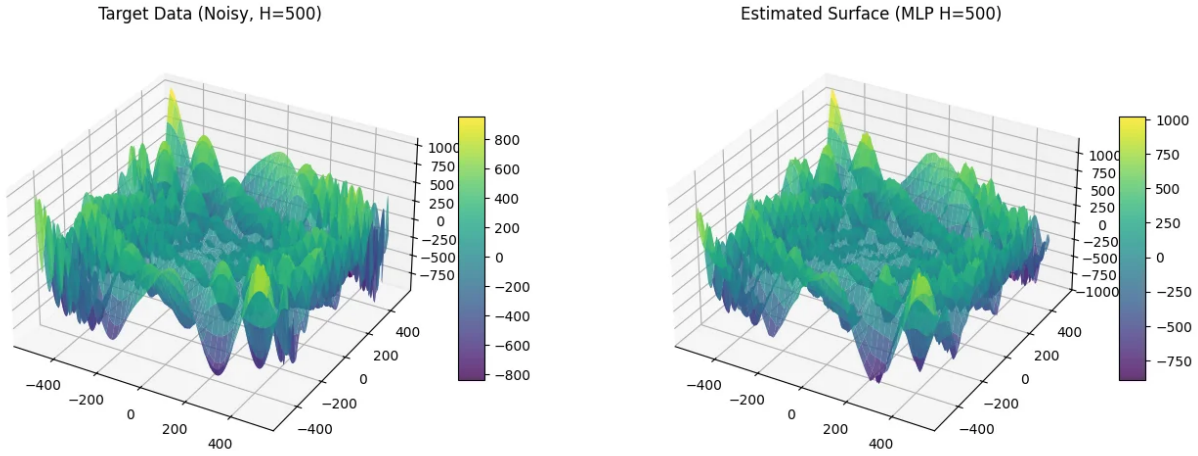


Figure 1: Target Data vs Estimated Surface (Baseline MLP, H=500)

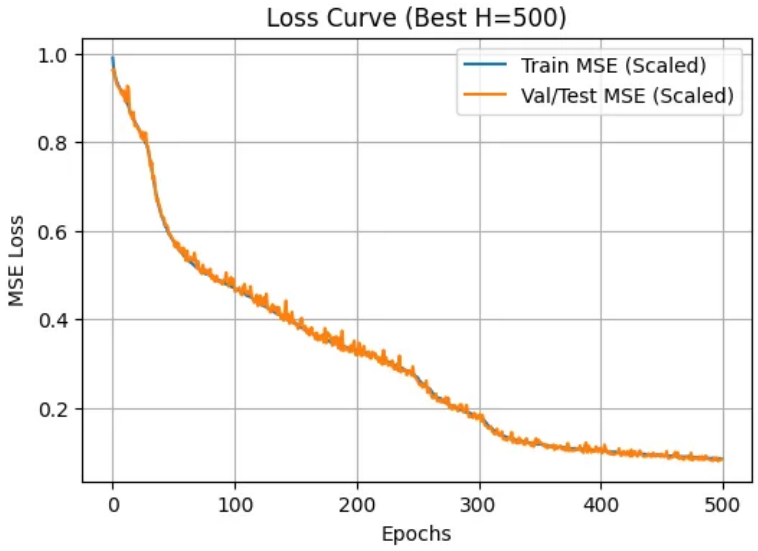


Figure 2: Loss Curve for Baseline MLP (H=500)

The model does not exhibit underfitting nor overfitting, as validation performance tracks training performance. It demonstrates good generalization, successfully approximating the complex target function.

(c) Test Error

	Optimal H	Err_{test}
Fold 1	500	0.3832
Fold 2	500	0.2896
Fold 3	500	0.2909
Mean±std	-	$\mathbf{0.3212 \pm 0.0537}$

(d) Analysis

The MLP with 500 hidden neurons achieves good approximation with 0.3212 Standardized RMSE (approx. 4.78% relative error). The model has sufficient capacity and demonstrates proper generalization with robust performance across folds. However, the error is higher than the theoretical noise floor, establishing a baseline for improvement.

Q1-7: Fourier Feature Mapping (Hand-designed Augmentation)

Rationale

```python
class FourierFeatureMapper:
    def __init__(self, input_dim=2, mapping_size=256, scale_factor=10):

        self.input_dim = input_dim
        self.mapping_size = mapping_size
        self.B = None
        self.scale_factor = scale_factor

    def fit(self, seed=42):
        np.random.seed(seed)
        b_low = np.random.normal(0, 1, size=(self.mapping_size // 4, self.input_dim))
        b_mid = np.random.normal(0, self.scale_factor, size=(self.mapping_size // 2, self.input_dim))
        b_high = np.random.normal(0, self.scale_factor * 10, size=(self.mapping_size // 4, self.input_dim))
        self.B = np.concatenate([b_low, b_mid, b_high], axis=0).T

    def transform(self, x):
        x_proj = (2. * np.pi * x) @ self.B
        x_sin = np.sin(x_proj)
        x_cos = np.cos(x_proj)
        x_aug = np.concatenate([x, x_sin, x_cos], axis=1)

        return x_aug
```

Figure 3: Fourier Feature Mapper Implementation

The target function $f(x)$ contains high-frequency sinusoidal components that standard MLPs struggle to learn. Fourier Feature Mapping transforms input coordinates into a higher-dimensional space using sine and cosine functions. We transform (x_1, x_2) from dimension 2 to $2+128+128 = 258$ dimensions using multi-scale frequency bands (low, mid, high).

(a) Nested CV Results

Table 3: 3-Fold CV Results with Fourier Features

# Hidden Neurons, H	10	100	200	500
Fold 1	0.1542 / 0.1583	0.0248 / 0.0294	0.0200 / 0.0274	0.0164 / 0.0210
Fold 2	0.1545 / 0.1575	0.0243 / 0.0295	0.0200 / 0.0370	0.0166 / 0.0214
Fold 3	0.1563 / 0.1593	0.0235 / 0.0294	0.0203 / 0.0255	0.0158 / 0.0202
Mean±std	0.1550 ± 0.0011 / 0.1584 ± 0.0009	0.0242 ± 0.0007 / 0.0294 ± 0.0001	0.0201 ± 0.0002 / 0.0300 ± 0.0062	$\mathbf{0.0163 \pm 0.0004}$ / $\mathbf{0.0209 \pm 0.0006}$

Best H = 500

(b) Visualization

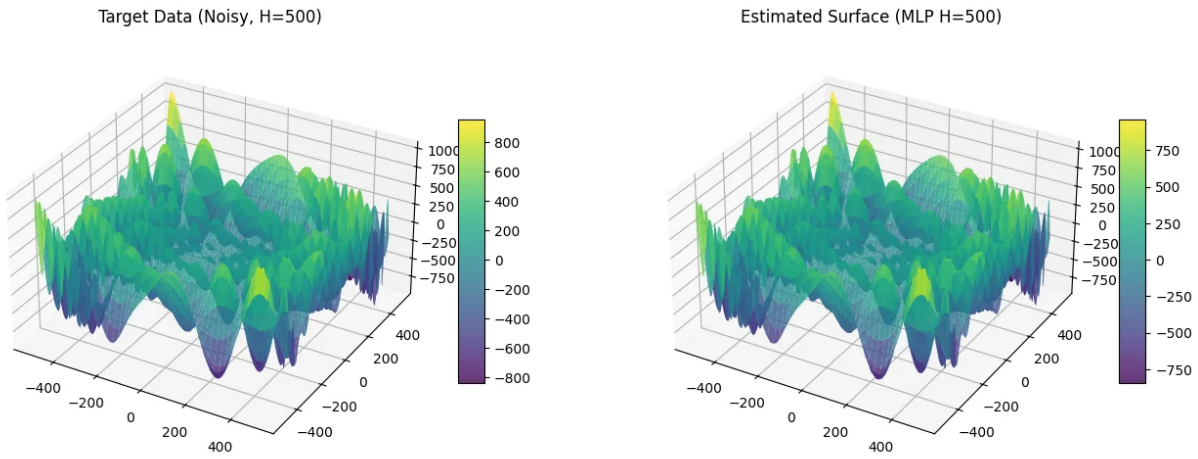


Figure 4: Target Data vs Estimated Surface (Fourier Features, H=500)

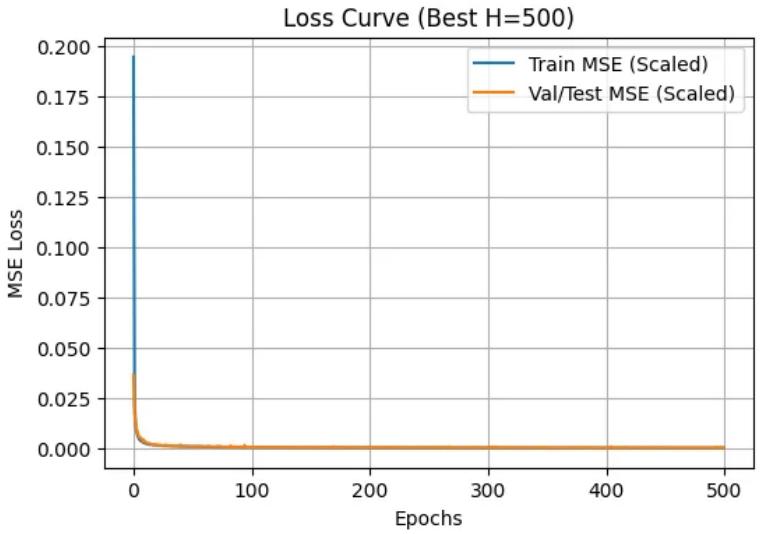


Figure 5: Loss Curve with Fourier Features (H=500)

The model exhibits excellent performance, with near-instant convergence and near-perfect train-validation alignment.

(c) Test Error

	Optimal H	Err_{test}
Fold 1	500	0.0183
Fold 2	500	0.0172
Fold 3	500	0.0168
Mean±std	-	$\mathbf{0.0174 \pm 0.0008}$

(d) Analysis

The MLP with Fourier features achieves 0.0174 Standardized RMSE (approx. 0.26% relative error), representing an **18.4× improvement** over the baseline. This confirms that properly designed input features can transform a difficult non-linear problem into an easily solvable task. The near-noise-level performance suggests the model has reached its asymptotic limit.

Q1-8: Three Hidden Layer MLP

Model Architecture

A three-hidden-layer MLP without augmented features. Candidates: $(10, 10, 10)$, $(100, 100, 100)$, $(200, 200, 200)$, $(500, 500, 500)$.

(a) Nested CV Results

Table 4: 3-Fold CV Results for 3-Layer MLP

# Hidden Neurons	(10,10,10)	(100,100,100)	(200,200,200)	(500,500,500)
Fold 1	0.4903 / 0.4904	0.0419 / 0.0396	0.0410 / 0.0440	0.0400 / 0.0347
Fold 2	0.4915 / 0.4876	0.0518 / 0.0525	0.0342 / 0.0479	0.0380 / 0.0407
Fold 3	0.4888 / 0.4884	0.0458 / 0.0501	0.0354 / 0.0485	0.0398 / 0.0386
Mean±std	0.4902 ± 0.0014 / 0.4888 ± 0.0014	0.0465 ± 0.0050 / 0.0474 ± 0.0069	$\mathbf{0.0369 \pm 0.0036}$ / $\mathbf{0.0468 \pm 0.0024}$	0.0393 ± 0.0011 / 0.0380 ± 0.0030

Best H = (200, 200, 200)

(b) Visualization

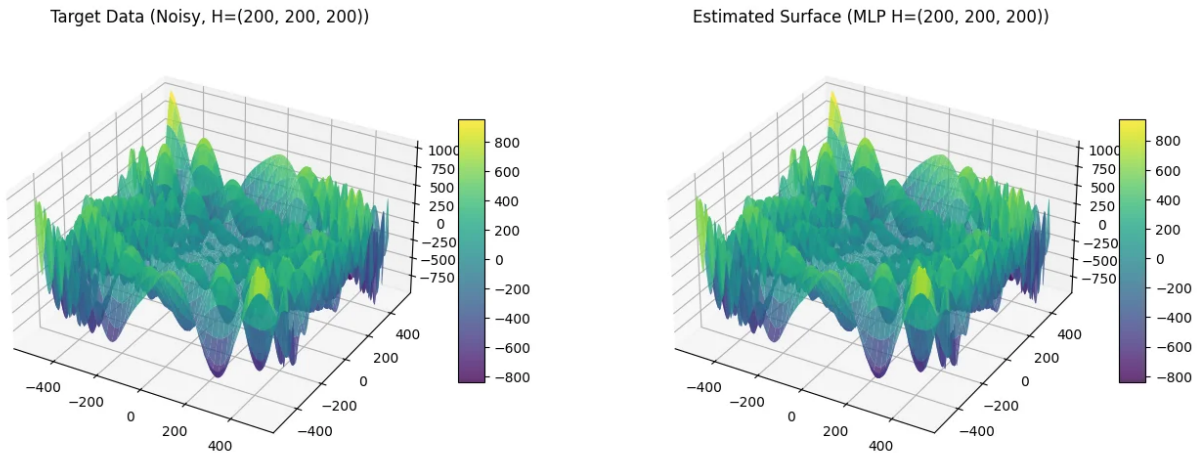


Figure 6: Target Data vs Estimated Surface (3-Layer MLP, H=(200,200,200))

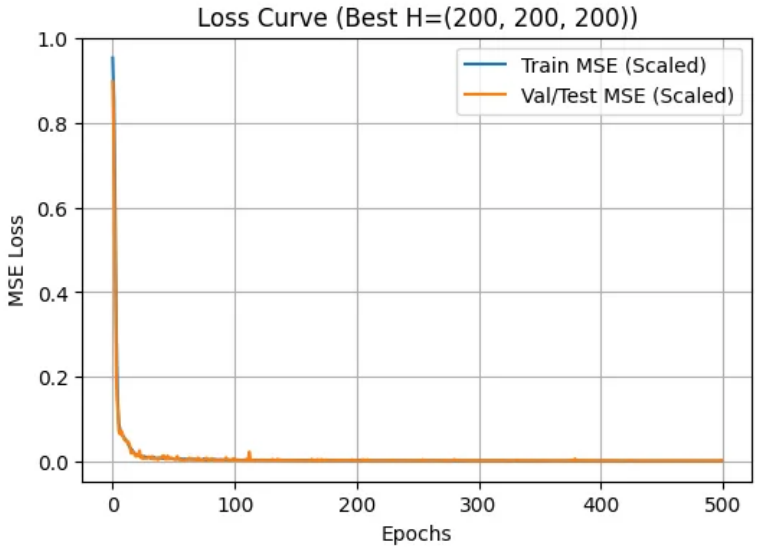


Figure 7: Loss Curve for 3-Layer MLP

(c) Test Error

	Optimal H	Err_{test}
Fold 1	(500,500,500)	0.0310
Fold 2	(200,200,200)	0.0319
Fold 3	(200,200,200)	0.0301
Mean±std	-	$\mathbf{0.0310 \pm 0.0009}$

(d) Analysis

The 3-layer MLP achieves 0.0310 Standardized RMSE (approx. 0.46% relative error), representing a **10.3× improvement** over the baseline. Depth increases representational power, allowing the network to learn hierarchical features. However, it still lags behind the Fourier feature model, highlighting that explicit domain knowledge often supersedes architectural depth.

Q1-9: Comparison of Three Models

Table 5: Comparison of Fitting, Generalization, and Optimization

Feature	**Model 1 (Q6)**	**Model 2 (Q7)**	**Model 3 (Q8)**
Architecture	Single-Layer (500)	Single-Layer (500)	Three-Layer (200-200-200)
Input Data	Raw Coordinates	Fourier Augmented	Raw Coordinates
Fitting Capability	Low (Underfitting)	Excellent (Near-Perfect)	High (Good Approximation)
Test RMSE (Std)	0.3212	**0.0174**	0.0310
Improvement	Baseline	18.4× vs Q6	10.4× vs Q6
Optimization	Stagnated at high loss	Fast & Easy convergence	Slower, requires depth

Fitting Capability

Model 2 (Fourier): Superior fitting. The Fourier features explicitly provide oscillatory basis functions needed for cusps and radial phases. **Model 3 (Deep):** Strong fitting through hierarchical feature learning. **Model 1 (Baseline):** Poor fitting due to inability to capture high-frequency variations.

Generalization Performance

Model 2 demonstrates the best generalization with minimal train-test gap and lowest std (±0.0008). Model 3 generalizes well but is 1.78× less accurate than Model 2. Model 1 has worst generalization, predicting only coarse global trends.

Optimization Difficulty

Model 2 is easiest—the Fourier features transform the problem into near-linear regression. Model 3 requires more epochs but converges successfully with proper initialization. Model 1 gets stuck in high-loss regions (optimization trap).

Conclusion

Model 2 (Augmented Features) is the best, illustrating that **"Data Representation > Model Architecture"**. A simple model with smart features outperforms a deep model with raw features. However, Model 3 demonstrates the power of Deep Learning when feature engineering is not possible.

Question 2: Image Dimension Reduction, Visualization, and Classification

Part A: Dimension Reduction with PCA

(a) PCA Projection Matrix

To construct the PCA projection matrix $W \in R^{n \times d}$, we use the entire training set of 50,000 CIFAR-10 images. The input dimension $n = 32 \times 32 \times 3 = \mathbf{3072}$. The output dimension $d = \mathbf{2}$ for 2D visualization. Thus, W maps from R^{3072} to R^2.

(b) 2D Feature Distribution

(a) Training Set (50,000 samples)

(b) Test Set (10,000 samples)

Figure 8: PCA 2D Feature Distribution for CIFAR-10

As shown, the feature points of different classes are heavily overlapped and do not form distinct clusters. PCA is an unsupervised linear method that maximizes variance without considering class labels, resulting in significant information loss when compressing 3072-dim to 2D.

(c) Softmax Regression Training

A Softmax Regression classifier (no hidden layers) maps 2D PCA features directly to 10 class scores. Trained for 50 epochs using Adam optimizer and Cross-Entropy Loss.

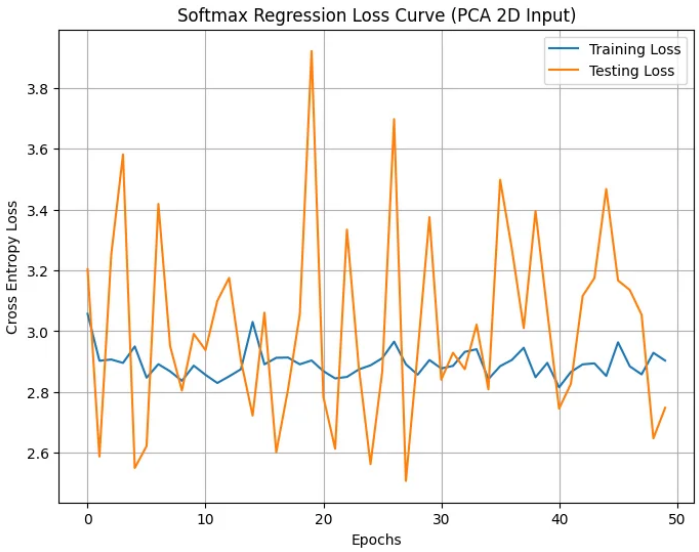


Figure 9: Softmax Regression Loss Curve (PCA 2D Input)

(d) Accuracy

Training Accuracy (%)	Testing Accuracy (%)
15.55	15.48

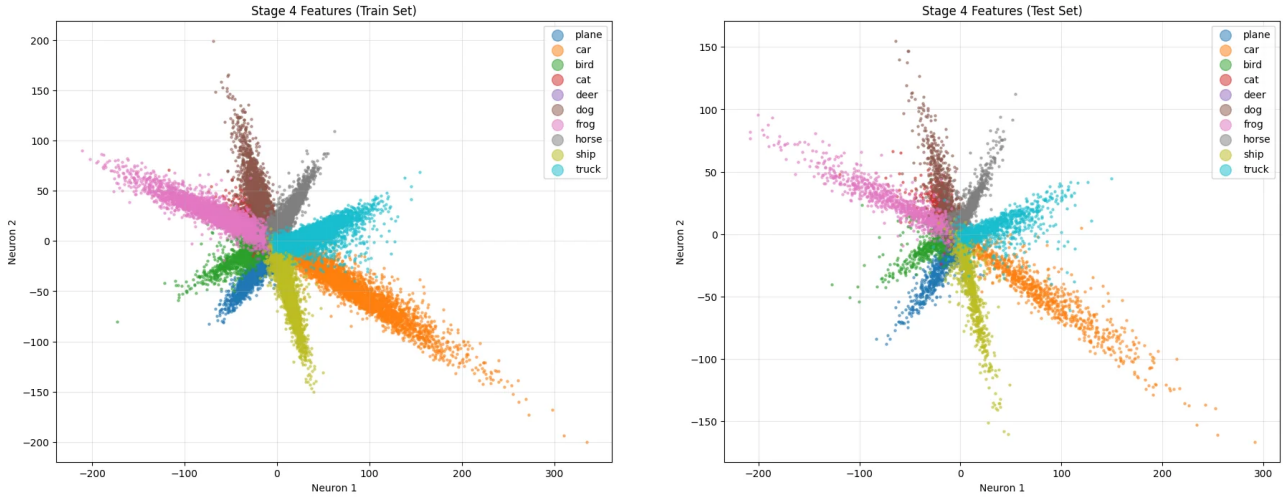
Part B: Dimension Reduction with CNN

Model Architecture & Hyperparameters

Table 6: CNN Architecture for Dimension Reduction

Component	Description
Preprocessing	ToTensor(), Normalize(mean=0.5, std=0.5)
Initialization	PyTorch default (Kaiming/He Uniform)
Optimizer	SGD (momentum=0.9, weight_decay=5e-4)
Loss	Cross-Entropy Loss
Learning Rate	Initial=0.01, StepLR (decay 0.1 every 10 epochs)
Batch Size	64
Epochs	15
Stage 1	2× Conv(5×5, 32) → ReLU → MaxPool(2×2)
Stage 2	2× Conv(5×5, 64) → ReLU → MaxPool(2×2)
Stage 3	2× Conv(5×5, 128) → ReLU → MaxPool(2×2)
Stage 4	Linear(2048 → 2) — *Dimension Reduction*
Stage 5	Linear(2 → 10) — *Output*

(a) Stage 4 Feature Distribution



(a) Training Set (50,000 samples)

(b) Test Set (10,000 samples)

Figure 10: CNN Stage 4 Feature Distribution for CIFAR-10

Unlike PCA, the CNN features show clear clusters corresponding to the 10 classes. Data points of the same class are grouped tightly, and different classes are well-separated. This demonstrates the network learned to map images into a 2D manifold where semantic separation is maximized.

(b) Softmax Regression on CNN Features

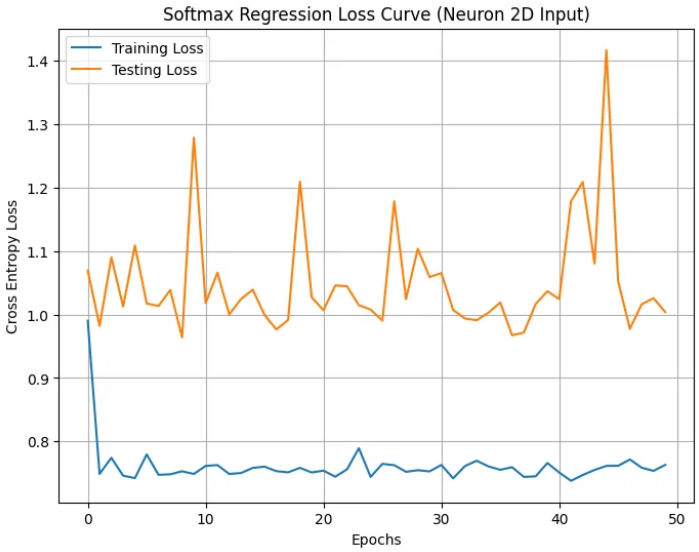


Figure 11: Softmax Regression Loss Curve (CNN 2D Input)

The CNN-based model demonstrates rapid loss decrease, proving that learned non-linear features are highly effective for classification.

(c) Accuracy

Training Accuracy (%)	Testing Accuracy (%)
74.98	71.42

(d) Comparison of Dimension Reduction Methods

Table 7: Comparison of PCA vs CNN for Dimension Reduction

Method	Learning Paradigm	Model Nature	Test Accuracy	Visualization
PCA	Unsupervised	Linear	15.48%	Poor (Overlapped)
CNN	Supervised	Non-linear	**71.42%**	Excellent (Clustered)

Learning Paradigm: PCA is unsupervised, focusing on variance preservation while ignoring class labels. CNN is supervised, guided by Cross-Entropy loss to separate classes.

Model Nature: PCA performs linear projection, unable to unfold complex non-linear image manifolds. CNN uses non-linear activations (ReLU) and deep layers to transform pixels into separable semantic representations.

(e) Best Model for Visualization

The CNN model is significantly better for visualization. While PCA preserves "most variable" directions, it fails to group semantically similar images. The CNN, trained with supervision, forces 2D features to be discriminative, resulting in scatter plots where each class forms a distinct cluster—allowing clear visual interpretation of dataset structure and class boundaries.

Question 3: Transfer Learning

(a) Modified ResNet18 Specification

The modified ResNet18 architecture is designed to handle 64×64 inputs without aggressive downsampling in the early stages. The stride of `conv1` and `maxpool` is set to 1. In other words, because the image size is nearly four times smaller than ImageNet, the reduction effect on the feature map size caused by the stride was scaled down by a factor of four, thereby adjusting the model to better fit the face dataset. In this process, to avoid losing the parameters already learned, the layers were not reconstructed and merged from scratch; instead, the existing layers were retained, and only their stride values were modified. The detailed specification of layers, weights, and receptive fields is presented in Table 8. In Table 8, following the instructions provided during the Q&A, the "# of weights" refers to the number of weights determined solely by the conv layers (excluding learnable parameters in layers such as Batch Normalization). Additionally, the weights from the 1×1 conv layers in the skip connections of Conv3, Conv4, and Conv5 were added to the counts of Conv3-1, Conv4-1, and Conv5-1, respectively.

Table 8: Modified ResNet18 Architecture Specification (Conv Weights Only)

Layer	**Stride/Padding**	**Feature Map Size**	**# Weights**	**Receptive Field**
Input	-	$64 \times 64 \times 3$	-	1
Conv1	S=1, P=3	$64 \times 64 \times 64$	9,408	7
Pooling	S=1, P=1	$64 \times 64 \times 64$	0	9
Conv2-1	S=1, P=1	$64 \times 64 \times 64$	36,864	11
Conv2-2	S=1, P=1	$64 \times 64 \times 64$	36,864	13
Conv2-3	S=1, P=1	$64 \times 64 \times 64$	36,864	15
Conv2-4	S=1, P=1	$64 \times 64 \times 64$	36,864	17
Conv3-1	S=2, P=1	$32 \times 32 \times 128$	81,920	19
Conv3-2	S=1, P=1	$32 \times 32 \times 128$	147,456	23
Conv3-3	S=1, P=1	$32 \times 32 \times 128$	147,456	27
Conv3-4	S=1, P=1	$32 \times 32 \times 128$	147,456	31
Conv4-1	S=2, P=1	$16 \times 16 \times 256$	327,680	35
Conv4-2	S=1, P=1	$16 \times 16 \times 256$	589,824	43
Conv4-3	S=1, P=1	$16 \times 16 \times 256$	589,824	51
Conv4-4	S=1, P=1	$16 \times 16 \times 256$	589,824	59
Conv5-1	S=2, P=1	$8 \times 8 \times 512$	1,310,720	67
Conv5-2	S=1, P=1	$8 \times 8 \times 512$	2,359,296	83
Conv5-3	S=1, P=1	$8 \times 8 \times 512$	2,359,296	99
Conv5-4	S=1, P=1	$8 \times 8 \times 512$	2,359,296	115
avgpool (GAP)	-	$1 \times 1 \times 512$	0	-

As shown in the table, all layers except the softmax layer retain the original input/output channels and kernel sizes, with only the stride values modified. Since the number of parameters in a layer depends on its input/output channels and kernel size, the total number of weights in the original ResNet18—excluding the softmax layer—remains unchanged.

(b) Baseline Model

We retrained the softmax layer of the pretrained ResNet18 while freezing all convolutional layers.

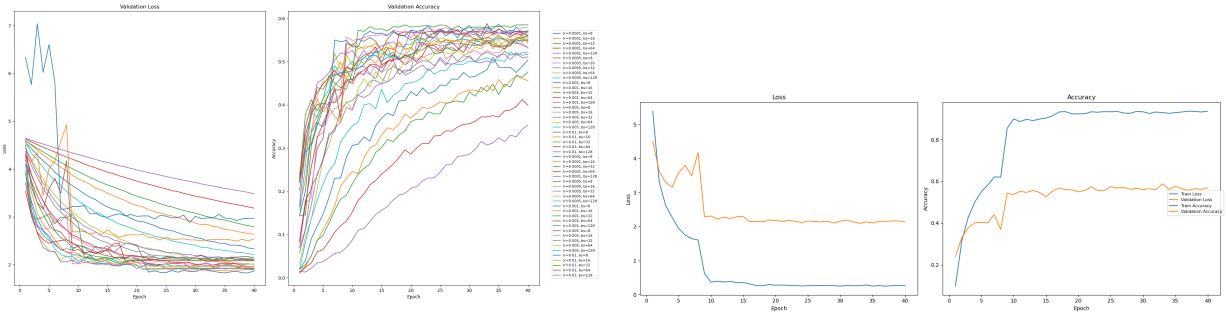
Dataset Construction

The validation set used for determining the reference loss of the ReduceLROnPlateau scheduler and for selecting hyperparameters was constructed in the same manner as in the grid search, by splitting the learning set into training and validation sets at an 8:2 ratio. Models A through D were configured in the same way; therefore, this will not be mentioned further.

Scheduler & Optimizer

- **Optimizer:** Adam
- **Scheduler:** ReduceLROnPlateau (factor=0.1, patience=3, threshold=1e-4)
  We chose ReduceLROnPlateau as our scheduler because it is relatively less sensitive to the number of epochs. Considering that the total number of epochs was 30, the scheduler was configured with a patience of 3 and a loss threshold of (1×10^{-4}), reducing the learning rate by a factor of 0.1 each time a plateau was detected. Models A through D were configured in the same manner; therefore, this will not be mentioned further.

Hyperparameter Selection



(a) Validation Loss & Accuracy of Grid Search Process

(b) Loss & Accuracy for Best Parameter Set

Figure 12: Grid Search Result

- Learning Rate = 0.005
- Batch Size = 8
- Epoch = 30

Based on the number of parameter updates, the batch-size candidates ([8, 16, 32, 64, 128]) were selected, and the learning-rate candidates ([1e-4, 5e-4, 1e-3, 5e-3, 1e-2]) were generated by scaling a base value of (1e-3) by factors of 10. A grid search was then performed over all batch–learning-rate combinations. The train dataset was split into a 0.8:0.2 ratio for train and validation, and the parameter set that achieved the highest validation accuracy within 40 epochs was selected. Figure 12-(a) is validation loss, accuracy graphs of whole parameter combinations, and (b) is the training, accuracy graph for the best parameter set. Based on the grid search results, the model configuration with batch size 8 and a learning rate of 0.005, which achieved the highest validation accuracy, was selected as the final choice.

With ReduceLROnPlateau scheduler, the validation loss often saturates or oscillates rather than decreasing explicitly, since the learning rate becomes very small in the later stages of training. Therefore, since saturation typically begins around 20 epochs in Figure 12-(b), all models were trained for a fixed 30 epochs. The model checkpoint saved at epoch 30 was used as the final model, but if a higher test accuracy appeared at an earlier epoch, that observation was also reported in the results for analysis.

Result

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
30	0.211	94.4	2.137	59.625	2.402	**56.4**
26 (Best)	0.230	93.6	2.129	59.75	2.35	**57.7**

Compared to ImageNet-1K, the face dataset used in this assignment is extremely small, and ImageNet-1K does not contain any classes that directly represent human faces—only indirect classes such as ball player(the similarity between the two datasets is low). Therefore, when only the classifier head was retrained while keeping the feature extractor fixed (Figure 13), the model achieved a training accuracy of 94.4%, whereas the test accuracy reached only 56.4% at epoch 30. Although the training process was reasonably successful considering the low similarity, the large gap between the training and test losses clearly indicates that severe overfitting occurred due to the limited size of the dataset.

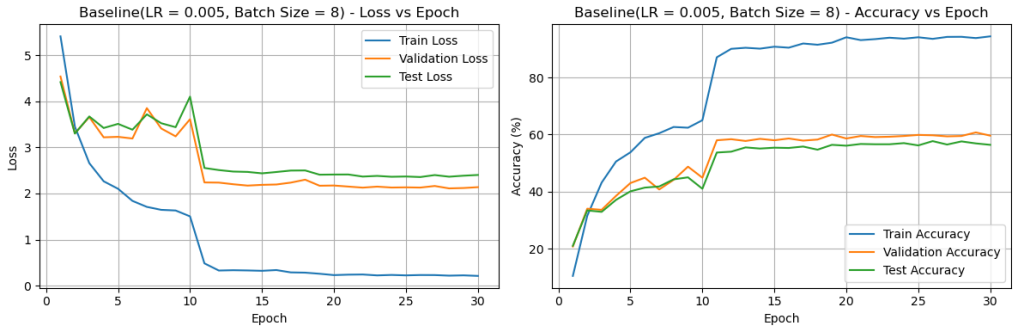


Figure 13: Baseline Training Curves (Test Acc: 56.4%)

(c) Model A: Fine-tune Conv5_x

We fine-tuned the `Conv5_x` block along with the FC layer.

Data Augmentation

- **RandomHorizontalFlip**: Probability=0.5
- **RandomAffine**: Degree=15, Translation=$(0.1, 0.1)$, Scale=$(0.9, 1.1)$
- **ColorJitter**: Brightness=0.2, Contrast=0.2, Saturation=0.2

Because the dataset is substantially smaller than ImageNet, an ablation study was conducted to identify the most effective combination of data augmentation techniques to improve validation performance. Since the convolutional layers were now partially included in training, the model could benefit from the translation invariance of CNN. Therefore, random affine transformations was applied. Additionally, upon examining the training dataset, it was observed that images within the same class exhibited large variations in color temperature, and some images were even in grayscale. To mitigate overfitting caused by RGB color bias, ColorJitter was used to add random color noise. Finally, because human faces are generally symmetric along the vertical axis, horizontal flipping was applied with a probability of 0.5.

The parameters used for each transformation are listed below. These settings were also applied to Model B; therefore, they will not be repeated in the description of Model B.

Hyperparameter Selection

- **Weight Decay**: 0.0001
  To suppress overfitting, the same weight decay was applied as a form of regularization across Models A through D.
- **Learning Rate**($LR = 0.005$ from Baseline): FC: $LR/10$, Conv5: $LR/10$
  Although the fully connected head is typically trained using the original learning rate, in this project the classification head had already been trained in the baseline model. Therefore, using (LR/10) instead of the full learning rate was considered more appropriate. In other words, an effective learning rate of 0.0005 was applied throughout training.

  For verification, Conv5 was fixed at (LR/10), and the FC layer was trained using both (LR) and (LR/10) for comparison. With LR, the validation loss and accuracy were 0.647 and 90.1%, respectively, whereas with LR/10, the validation loss and accuracy were 0.459 and 90.6%. Although the accuracy with (LR/10) was only slightly higher within a reasonable margin of variation, the validation loss was significantly lower. This confirms that (LR/10) is the more appropriate choice.

Result

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
30	0.052	98.5	0.459	90.6	0.455	**90.8**
28(Best)	0.053	98.5	0.502	89.8	0.476	**90.9**

In contrast to the severe overfitting observed in the baseline model, the test accuracy increased substantially to around 90%, indicating that the model had been effectively regularized and generalized. Moreover, even

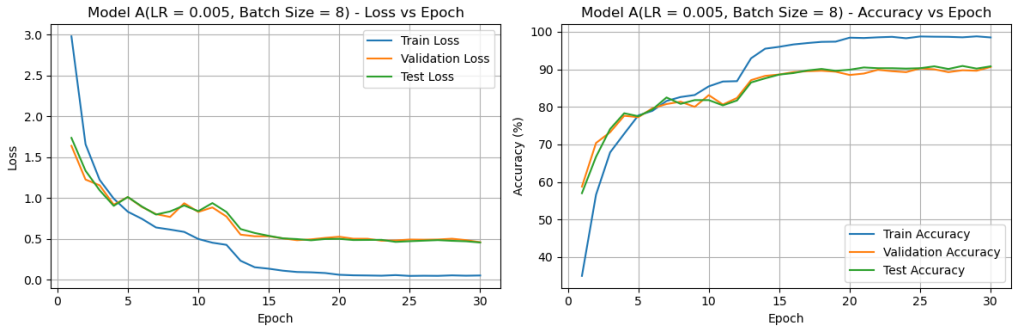


Figure 14: Model A Training Curves (Test Acc: 90.8%)

the training accuracy improved, suggesting that the learning process itself also benefited from the applied regularization.

(d) Model B: Fine-tune Conv4_x and Conv5_x

We fine-tuned both `Conv4_x` and `Conv5_x` blocks.

Data Augmentation

The augmentation settings applied to Model A were identically applied here as well.

Hyperparameter Selection

- **Weight Decay**: 0.0001
- **Learning Rate**($LR = 0.005$ from Baseline): FC: $LR/10$, Conv5: $LR/10$, Conv4: $LR/10$
  Since training was extended to earlier convolutional layers, we conducted an experiment to determine an appropriate boundary between the intermediate and later feature extractor by applying LR/10 and LR/100 to Conv4, respectively. The model using LR/10 achieved a validation loss and accuracy of 0.284 and 93.4%, whereas the model using LR/100 produced values of 0.442 and 90.1%. Because the model trained with LR/10 showed significantly better performance in both validation loss and accuracy, LR/10 was selected. This suggests that, given the substantial difference in similarity between the datasets, reducing the learning rate further at this stage was not yet appropriate.

Result

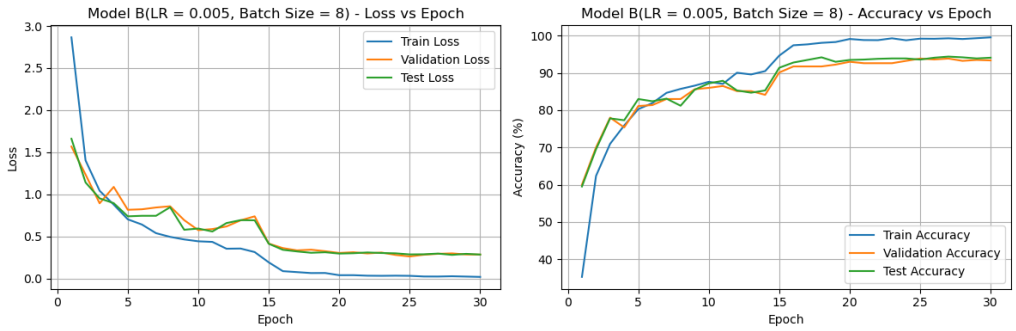


Figure 15: Model B Training Curves (Test Acc: 94.10%)

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
30	0.019	99.6	0.284	93.375	0.285	**94.1**
27(Best)	0.024	99.3	0.293	93.875	0.296	**94.4**

Since more convolutional layers were trained, the model was able to learn facial features more effectively, resulting in a slight increase in training accuracy. The test accuracy also improved noticeably, achieving a higher accuracy of 93.4

(e) Model C: Fine-tune All Layers

We fine-tuned all convolutional layers using differential learning rates to prevent destroying low-level features.

Data Augmentation

- **RandomHorizontalFlip**: Probability=0.5
- **RandomAffine**: Degree=15, Translation=$(0.1, 0.1)$, Scale=$(0.9, 1.1)$
- **ColorJitter**: Brightness=0.2, Contrast=0.2, Saturation=0.2

Since the entire model was now being trained, its capacity increased slightly, and thus stronger data augmentation was considered to further improve generalization performance. The horizontal flip was deemed appropriate as it already flips half of the images, but the strengths of ColorJitter and RandomAffine were slightly increased for this experiment.

Setting	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)
Same with Model A, B	0.018	99.5	0.304	93.0
More Color Jitter	0.021	99.3	0.292	93.9
More Affine Transform	0.026	99.3	0.280	93.1
Both of Them	0.025	99.2	0.397	92.0

In the 'More Color Jitter' setting, the brightness, contrast, and saturation parameters were each increased to 0.3. For 'More Affine Transform', stronger augmentation was applied with Degree=20, Translation=$(0.15, 0.15)$, Scale=$(0.85, 1.15)$. In the 'Both of Them' setting, both augmentation configurations were applied simultaneously.

Among the three options, the model with the lowest validation loss at epoch 30 was 'More Affine Transform', while the model with the highest validation accuracy was 'More Color Jitter'. Considering the overall training behavior, the difference between validation losses (0.292 vs. 0.28) falls within a reasonable margin of variation, whereas the difference in validation accuracy (93.9% vs. 93.1%) is more substantial. Therefore, the data augmentation settings for Models A and B were updated by strengthening only the Color Jitter component.

Hyperparameter Selection

- **Weight Decay**: 0.0001
  To suppress overfitting, the same weight decay was applied as a form of regularization across Models A through D.
- **Learning Rate**(LR = 0.005 from Baseline): FC: $LR/10$, Conv5 - Conv4: $LR/10$, Conv3 - Conv1 & Stem: $LR/100$
  Based on the results from Model B, the settings for the FC layer through Conv4 were kept the same. For Conv3 through the stem, a lower learning rate was applied so that the coarse feature extractors would be updated more slowly during training.

  First, an experiment was conducted to determine whether LR/100 should be applied starting from the Conv3 layer. The model trained with LR/10 achieved a validation loss and accuracy of 0.305 and 93.4%, respectively, whereas the model trained with LR/100 achieved 0.292 and 93.9%. Since the performance was better with (LR/100), the Conv3 layer and all earlier layers were treated as earlier or intermediate convolutional layers and trained with (LR/100).

  To determine whether LR/100 or LR/1000 should be applied to the stem layer, an additional experiment was conducted. The model trained with LR/100 achieved a validation loss and accuracy of 0.292 and 93.875%, respectively, whereas the model trained with LR/1000 recorded values of 0.321 and 92.625%. Since the performance with LR/100 was clearly better, LR/100 was selected.

  This result contradicts the intuition that the stem, which extracts more coarse features, should require a smaller learning rate. To investigate the cause, we revisited the earlier modification of ResNet18, where the stride of the stem was modified while keeping its weights unchanged. This caused a slight mismatch between the pretrained stem weights and the modified model structure, and during Model C, the stem weights were adjusted during training, leading to the observed outcome.

Result

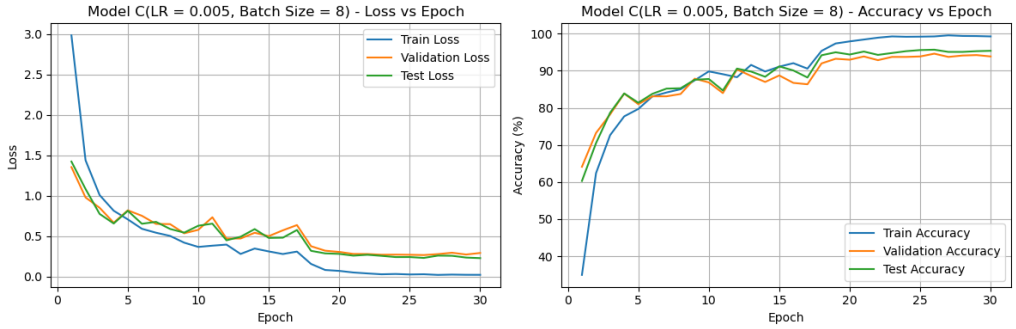


Figure 16: Model C Training Curves (Test Acc: 95.4%)

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
30	0.021	99.3	0.292	93.9	0.228	**95.4**
26(best)	0.028	99.3	0.266	94.6	0.231	**95.7**

With the applied hyperparameters and regularization strategies functioning effectively, the model achieved the highest test accuracy of 95.4

(f) Model D: MLP Head

We froze all convolution blocks and added 2-hidden-layer MLP to the simple linear classifier ($512 \to 1024 \to 512 \to 100$) with BatchNorm, ReLU, and Dropout.

Data Augmentation

Since significant overfitting had already been observed during the training of the baseline model, it was reasonable to expect that a similar issue would arise in the model with additional MLP layers. Therefore, an ablation study was conducted to determine which data augmentation techniques the MLP head could effectively accommodate.

Because an MLP does not naturally possess the translation invariance characteristic of CNNs, RandomAffine was removed. Accordingly, additional experiments were performed using Horizontal Flip and Color Jitter. The parameters for each transformation followed the same settings used in Models A and B.

Augmentation	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)
No Augmentation	0.137	96.4	1.784	60.3
Horizontal Flip	0.391	89.3	1.827	59.6
Color Jitter	0.302	92.1	1.810	61.1
Both of Them	0.529	85.2	1.562	65.0

When Horizontal Flip and Color Jitter were applied individually, the improvements were either within the margin of error or only slightly better, and in fact the validation loss increased. However, applying both augmentations together resulted in a meaningful improvement. Therefore, even though training performance decreased, the additional data augmentation was adopted.

Hyperparameter Selection

- **Hyperparameters:** , Dropout=0.1, weight decay=0.0001.
- **Scheduler:** ReduceLROnPlateau (factor=0.1, patience=3).
- **Result:** The model suffered from overfitting, showing high training accuracy but poor testing accuracy compared to CNN fine-tuning.

– **Weight Decay**: 0.0001
To suppress overfitting, the same weight decay was applied as a form of regularization across Models A through D.

– **Dropout**: Inactive Probability $p = 0.1$ When a stronger regularization of 0.2 or higher was applied, even the training process did not proceed properly. Therefore, a mild dropout rate of 0.1 was used to alleviate overfitting.

– **Learning Rate**($LR = 0.005$ from Baseline): New MLP layers LR, Pretrained head LR/10
Based on the results from Model A, the pretrained head from the baseline model was reused as the final output layer, and a lower learning rate(LR/10) was applied to it. The two newly added layers placed before the output layer, however, needed to be trained from scratch, and thus the original learning rate LR was applied to them.

Result

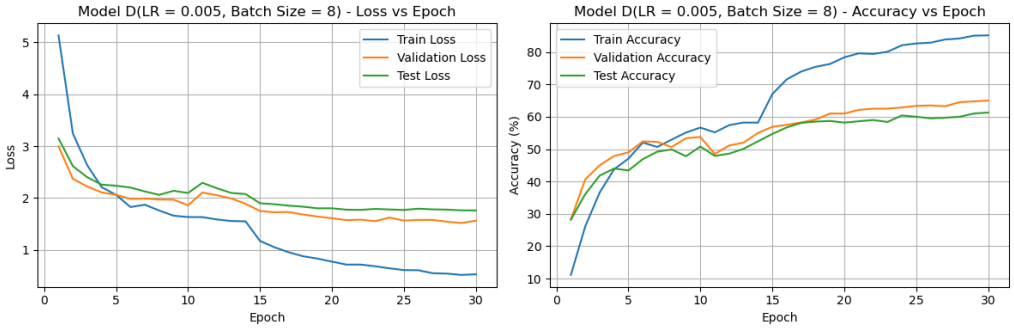


Figure 17: Model D Training Curves (Test Acc: 61.3%)

Epoch	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
30(best)	0.529	85.2	1.562	65.0	1.759	**61.3**

Although overfitting was not fully resolved, the results show a meaningful improvement compared to the baseline. However, due to the inherent limitations of training only the MLP, it is unlikely to achieve the level of performance observed in Models A–C.

(g) Analysis and Performance Summary

In this assignment, a ResNet18 model pretrained on ImageNet-1K is applied to a downstream task such as face recognition. The provided dataset consists of 4,000 training images, 1,000 test images, and 100 classes—significantly smaller than a large-scale dataset like ImageNet-1K. Moreover, ImageNet-1K does not contain classes directly related to human faces, resulting in low similarity between the two datasets. Thus, this task corresponds to **Quadrant 3**. Consequently, aside from the early convolutional layers, most layers require fine-tuning, and extensive data augmentation must be employed to mitigate overfitting.

From the perspective of the Size–Similarity matrix, Model B fine-tunes slightly few layers, whereas Model C updates the entire network. Therefore, it is reasonable to expect that Models B and C would achieve the highest performance.

- **Fine-tuning Efficacy:** Models A, B, and C significantly outperformed the Baseline. This confirms that while ImageNet features are useful, the domain shift (Objects → Faces) and the resolution change (224 → 64) require adaptation of the conv filters.

- **Baseline (Overfitting):** Training only the final FC head corresponds to a Quadrant 4 scenario, where the dataset is small and highly similar to the pretraining domain. However, in this assignment the dataset has low similarity to ImageNet, so although training performance was not poor, the model exhibited severe overfitting and poor test performance.

- **Model A:** Fine-tuning only the later conv layers corresponds to a Quadrant 1 setting, where the dataset is relatively large and similar to the pretraining domain. With more conv layers being updated, the model began to learn facial features more effectively. Combined with data augmentation and regularization

Table 9: Parameters & Performance Summary

Model	Learning rate and Hyperparameters	Training Acc.	Testing Acc.
Common Settings	**Learning Rate** LR = 0.005, **Batch Size** = 8, **Epoch** = 30 **Scheduler**: ReduceLROnPlateau (factor=0.1, patience=3, threshold=1e-4) **Optimizer**: (Baseline) Adam w/o weight decay, (Models A–D) Adam w/ weight decay = 0.0001	–	–
Baseline	Common Settings Only	94.4%	56.4%
Model A	**Data Augmentation**: HorizontalFlip(p=0.5), Affine(degree=15, translation=(0.1, 0.1), scale=(0.9, 1.1)), ColorJitter(brightness=contrast=saturation=0.2) **Learning Rate**: LR/10 for FC & Conv5	98.5%	90.8%
Model B	**Data Augmentation**: HorizontalFlip(p=0.5), Affine(degree=15, translation=(0.1, 0.1), scale=(0.9, 1.1)), ColorJitter(brightness=contrast=saturation=0.2) **Learning Rate**: LR/10 for FC & Conv5–Conv4	**99.6%**	94.1%
Model C	**Data Augmentation**: HorizontalFlip(p=0.5), Affine(degree=15, translation=(0.1, 0.1), scale=(0.9, 1.1)), ColorJitter(brightness=contrast=saturation=0.3) **Learning Rate**: LR/10 for FC & Conv5–Conv4, LR/100 for Conv3–Conv1 & Stem	99.3%	**95.4%**
Model D	**Data Augmentation**: HorizontalFlip(p=0.5), ColorJitter(brightness=contrast=saturation=0.2) **Dropout rate**: 0.1 **Learning Rate**: LR/10 for output layer, LR for 2 hidden layers **Nodes**: 1st hidden layer = 1024, 2nd hidden layer = 512, output layer = 100	85.2%	61.3%

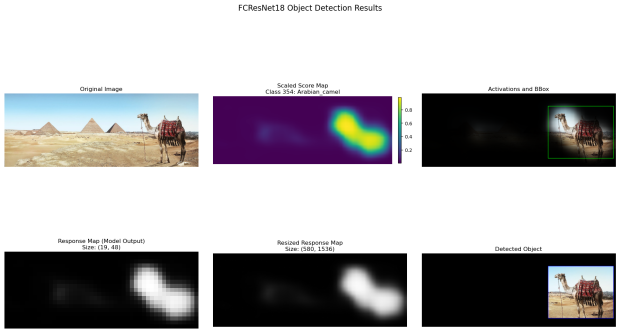
(weight decay, etc.), Model A achieved substantially higher performance than the baseline, although a small amount of overfitting remained when comparing training and test accuracy.

- **Model B:** This model corresponds to a Quadrant 3 scenario, which closely matches the conditions of this assignment. By fine-tuning one additional mid-level conv block, the model was able to adapt intermediate features to the face domain, resulting in further performance improvement. However, it does not appear to be the most appropriate model for a true Quadrant 3 scenario, and additional fine-tuning of more intermediate convolutional layers would likely be necessary.

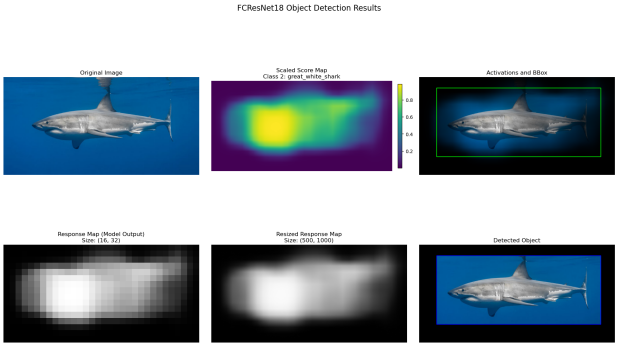
- **Model C (Best):** Fine-tuning the entire network with carefully scaled learning rates yielded the best results, enabling successful adaptation of both low-level and high-level feature extractors. Although full-network fine-tuning is typically suitable for Quadrant 2 (large dataset, low similarity), Model C still achieved the highest performance. Strong data augmentation mitigated issues arising from the small dataset size, and the initial mismatch from modifying stride of the stem layer without retraining was corrected during full fine-tuning, contributing to the performance gain.

- **Model D (Overfitting):** Replacing the classifier with a larger MLP increased the parameter count without improving feature extraction, resulting in severe overfitting in which the model memorized the training set but failed to generalize. Nonetheless, with multiple regularization techniques (dropout, BN layers, data augmentation, weight decay), Model D still achieved meaningful improvements over the baseline.

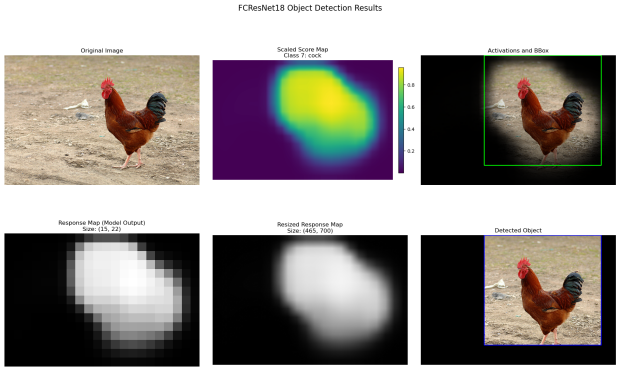
Question 4: Training-free Object Detector



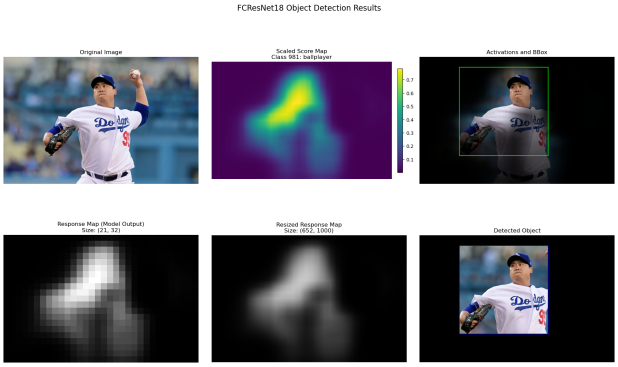
(a) Arabian Camel (Class 354)



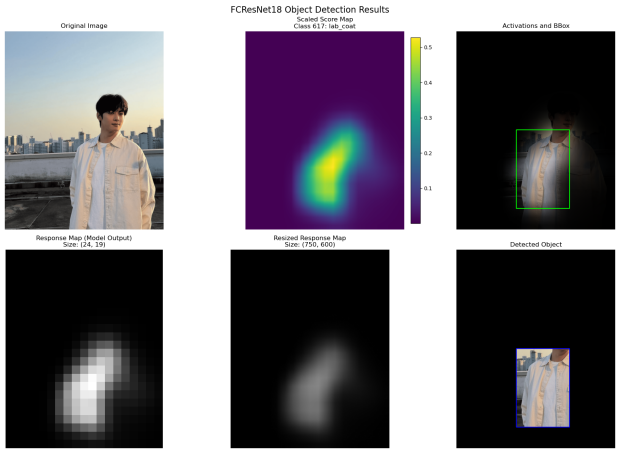
(b) Z: Great White Shark (Class 2)



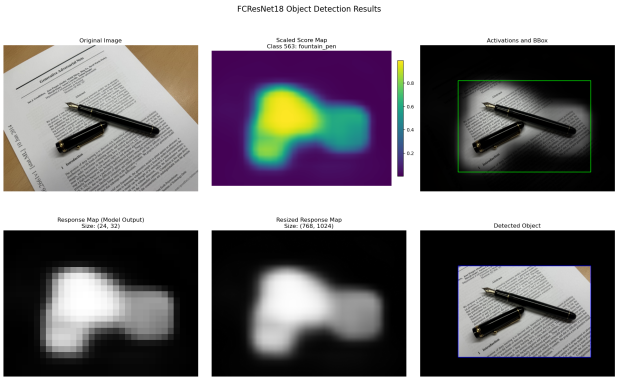
(c) Z+5: Cock (Class 7)



(d) Face 1: Ballplayer (Class 981) - F Success



(e) Face 2: Failure (Class 617: Lab Coat) - Unseen Class



(f) Random 1: Fountain Pen (Class 563)

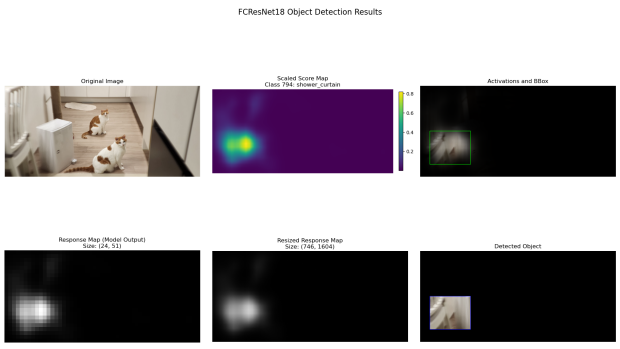
Figure 18: Detection results for the Standard, Student ID, and Face/Random tasks. Note the detection failure on the unseen class, as shown in the Cha Eun-woo case (e).

Convert ResNet18 to FCResNet18

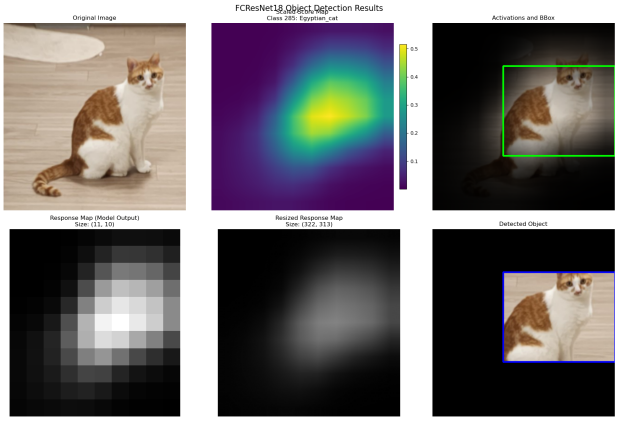
To design the training-free object detector, the GAP and FC head of the pretrained ResNet18 were modified. The Global Average Pooling layer, originally used to make the network independent of input image size, was removed, and replaced with an average pooling layer with a 7×7 kernel (stride 1, padding 3), allowing the output feature map of the final conv layer to retain its spatial dimensions of $n \times m \times 512$

Furthermore, the input-image-size–dependent FC layer was replaced with a 1×1 conv layer so that, instead of producing a $1 \times 1 \times 1000$ output vector for the 1000 classes, the model outputs a probability map of size $n \times m \times 1000$.

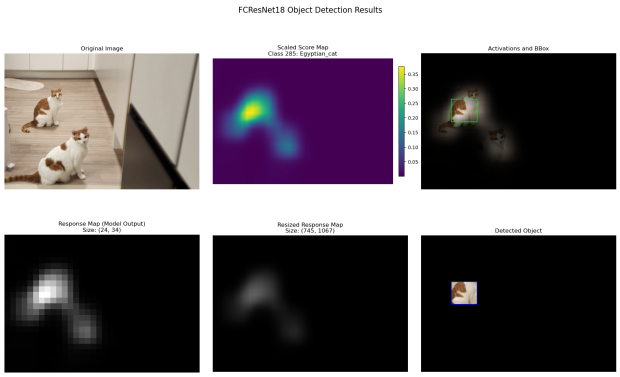
The original FC layer's weights(512×1000 matrix) can be reinterpreted as a $1000 \times 512 \times 1 \times 1$ tensor, corresponding to a 1×1 convolution with 512 input channels and 1000 output channels. By reusing the FC



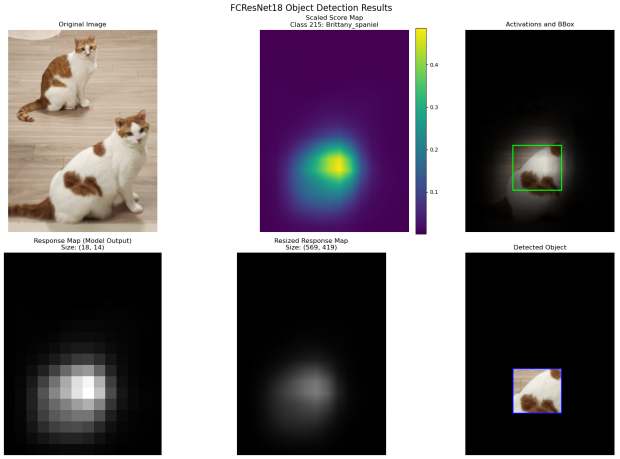
(a) Random2: Full Image Failure (Class 794: Shower Curtain)



(b) Crop 1: Egyptian Cat (Class 285)



(c) Crop 2: Egyptian Cat (Class 285)



(d) Crop 3: Brittany Spaniel (Class 215)

Figure 19: Analysis of the multi-object scene. The model failed on the full image (a) due to background texture but successfully detected objects when cropped (b-d).

weights in this way, the classification head's representational ability is preserved.

Inference & Visualization Results

Figure 18 shows the detection results obtained using FCResNet18 for the following images: (a) the provided Arabian Camel image; (b) and (c) the class images corresponding to the last three digits of the student ID, Z (2020132002 → 002), and Z+5; (d) an image containing a face that corresponds to an ImageNet class, Ballplayer (Ryu Hyun-jin); (e) an image containing a face with no similar class in ImageNet, Cha Eun-woo(Entertainer); and (f) a Random 1 image (taken with my phone);

Thresholding Strategy

To overlay the response map on the input image, the class-specific score map was first upsampled to match the input resolution using bilinear interpolation. The resulting heatmap was then binarized using a threshold value of **0.25** (i.e., 25% of the maximum response value).

This threshold was selected empirically by observing the trade-off between precision and recall:

- **Lower thresholds (<0.2):** Include excessive background regions, resulting in oversized bounding boxes that fail to tightly localize the object.
- **Higher thresholds (>0.4):** Only capture the most discriminative parts (e.g., the head of an animal), missing significant portions of the object body.
- **Threshold = 0.25:** Provides a balanced detection that captures the majority of the object while excluding most background noise, yielding tight yet complete bounding boxes across diverse object categories.

Figure 19 likewise presents the detection results (a) for Random 2 image , which was my friend's cats(Rata & Touille). However, since the model failed to detect any class when the full image was used, three additional images were created by cropping the original from different viewpoints, and detection was performed on these cropped images (b–d).

Q4-6. Observations on Experimental Result, Detector's Limitations

Analysis of Arabian Camel, Z, Z+5, Random 1 Images

The successful detections in Figures 18(a–c, f) demonstrate how CAM-based localization leverages class-discriminative features learned during ImageNet pretraining.

Response Map Interpretation: The response map highlights spatial regions that contribute most strongly to the predicted class. For each pixel location (i, j), the response value represents the weighted sum of feature activations at that location, where weights correspond to the importance of each channel for the target class. High-response regions indicate where the network "attends" to make its classification decision.

Why Localization Works:

- **Spatial Preservation:** By replacing GAP with a 7×7 average pooling layer (stride 1, padding 3), the FCResNet18 retains spatial correspondence between feature maps and input pixels. This allows the response map to preserve the object's location rather than collapsing it into a single prediction vector.
- **Discriminative Feature Learning:** During ImageNet training, the network learns to recognize class-specific visual patterns that distinguish one category from another. These learned features produce strong activations across the entire object region, enabling accurate localization of the full object extent.

Per-Image Analysis:

- **Arabian Camel (Fig. 18a):** The response map successfully captures the entire camel silhouette, with strong activations distributed across the body. The desert background produces minimal activation, enabling clean foreground-background separation.
- **Great White Shark (Fig. 18b):** The network activates across the shark's entire body contour. The high contrast between the shark and the uniform blue water background facilitates precise localization.
- **Cock (Fig. 18c):** The response map covers the full extent of the bird's body, accurately capturing its shape while suppressing background foliage.
- **Fountain Pen (Fig. 18f):** The model produces a well-distributed response along the entire length of the pen, successfully distinguishing it from the surrounding surface.

Analysis of Face Images: Contextual Learning vs. Unseen Class

- **Contextual Success (Fig. 18d):** The detection of 'Ballplayer' for Ryu Hyun-jin demonstrates how the model leverages **contextual cues** for classification. Although ImageNet-1k lacks a generic 'Face' or 'Person' category, the model successfully localized the player by recognizing associated objects and context (e.g., striped uniform, pitching posture, baseball glove). This indicates that the pretrained network encodes rich semantic associations beyond isolated object features.
- **Unseen Class Failure (Fig. 18e):** In contrast, the portrait of Cha Eun-woo was misclassified as 'Lab Coat' (Class 617), even when tested at lower resolutions. This failure is not due to texture bias but rather a **zero-shot classification problem**—ImageNet-1k contains no class that directly represents a human face or person. Without contextual objects (unlike the baseball scenario), the model defaulted to the most salient visual feature: the clothing texture. This comparison highlights the critical role of **dataset-task similarity**; when the target class is absent from the training taxonomy, the model cannot generalize regardless of image resolution.

Analysis of Random 2 Images: Ablation Study on Multi-object Scene

We performed a step-by-step ablation study to investigate the 'Shower Curtain' misclassification and localization errors shown above.

1. Ablation Steps and Results

- **(a) Original Image (Blurry Edges → Shower Curtain):** The background blur (bokeh) created vertical gradient lines. The texture-biased CNN interpreted these lines as the folds of a ' Shower Curtain', ignoring the foreground cats.
- **(b) Single Cat Crop (Scale Normalization):** Cropping to a single cat corrected the classification (Egyptian Cat), but the **Bounding Box (BBox)** remained inaccurate and diffuse.
- **(c) Blur Removal (Two Cats):** Removing the background blur fixed the texture bias issue (detected 'Cat'). However, the model failed to separate the two instances, creating a merged heatmap blob.
- **(d) Tight Crop (Two Cats → Brittany Spaniel):** Even with a tight crop, the model could not distinguish the two individual cats, treating them as a single texture distribution.

2. Root Cause Analysis

1. **Local Receptive Field:** The model sees local patches (fur, background lines) but fails to integrate them into a global shape due to the high resolution.
2. **Lack of Instance Awareness:** Without an RPN (Region Proposal Network), CAM-based methods cannot separate multiple instances of the same class.
3. **Limited Taxonomy:** ImageNet-1k lacks complex categories like 'group of cats'.

3. Proposed Solutions

To address the limitations observed in both Fig. 18 and Fig. 19, we propose:

- **Global Context (ViT):** Adopting Vision Transformers to capture global shape and context, reducing texture bias.
- **High-Res Adaptation:** Using Multi-scale training or Swin Transformers to handle high-resolution inputs without losing global coherence.
- **Instance Detection Heads:** Implementing explicit object detection heads (e.g., Faster R-CNN, YOLO) to handle multi-object scenarios and precise localization.

Experimental Results

Table 10: Object Detection Results

Class ID	Class Label	Map Size	Input Image	Detection
354 (Ex.)	Arabian Camel	19×48	580×1536	Success
002 (Z)	Great White Shark	16×32	500×1000	Success
007 (Z+5)	Cock	15×22	465×700	Success
981 (Face)	Ballplayer	21×32	652×1000	Success
608 (Face)	Lab Coat	24×19	750×600	Failure (Unseen Class)
563 (Random1)	Fountain Pen	24×32	768×1024	Success
Multi-object Scene Analysis (Two Cats)				
794 (Random2)	Shower Curtain*	24×51	746×1604	Failure (Wrong Class)
285	Egyptian Cat (Crop 1)	11×10	322×313	Success
285	Egyptian Cat (Crop 2)	24×34	745×1067	Success
215	Brittany Spaniel (Crop 3)	18×14	569×419	Failure (Wrong Class)

Discussions

Question 1

The experiments in Question 1 reveal a critical trade-off between model architecture and feature engineering when approximating complex, high-frequency functions. The baseline single-layer MLP (Q6) failed to capture the target surface's rapid oscillations, resulting in significant underfitting (Standardized RMSE ~0.32) even with 500 neurons. This indicates that a shallow network operating on raw coordinates lacks the inductive bias necessary to model high-frequency sinusoidal components.

In contrast, the introduction of Fourier Feature Mapping (Q7) yielded the best performance (RMSE ~0.017), improving accuracy by over 18x compared to the baseline. By explicitly transforming inputs into a higher-dimensional frequency domain, the non-linear problem was effectively simplified, allowing a shallow network to converge rapidly to a near-perfect solution. The deep three-layer network (Q8) also achieved strong performance (RMSE ~0.031) by autonomously learning hierarchical representations, but it still lagged slightly behind the feature-engineered model. This demonstrates that while deep architectures can approximate complex functions through composition, explicit domain knowledge (feature engineering) often provides a more efficient and precise path to convergence than architectural depth alone.

Question 2

The comparison in Question 2 highlights the fundamental difference between unsupervised linear projection and supervised non-linear learning for image data. PCA (Part A) performed poorly, yielding indistinguishable clusters and a classification accuracy of only ~15.5%. As an unsupervised linear method, PCA minimizes information loss based on global pixel variance but fails to preserve class-discriminative structures, causing different classes to overlap heavily in the reduced 2D space.

Conversely, the CNN-based approach (Part B) achieved distinct, well-separated clusters and a significantly higher accuracy of ~71.4%. The success of the CNN stems from its non-linearity and supervision. The non-linear layers (ReLU, Convolution) allow the model to "unfold" the complex, high-dimensional image manifold, while the supervised loss function forces the network to learn a 2D representation that maximizes class separability rather than just variance. This confirms that for semantic tasks like classification, dimension reduction must be guided by supervised signals to retain meaningful class boundaries.

Question 3

This assignment involves fine-tuning a ResNet18 model pretrained on ImageNet-1K and applying it to a downstream task: face recognition. The provided dataset consists of 4,000 training images, 1,000 test images, and 100 classes. It is much smaller than ImageNet-1K. Furthermore, ImageNet-1K contains no classes directly related to human faces, resulting in low similarity between the two datasets. Therefore, this task corresponds to **Quadrant 3** of the Size–Similarity Matrix.

Baseline & Model D

Both the Baseline and Model D attempted to solve the task by fine-tuning only an MLP head. Due to the substantial difference in similarity between the datasets and the limited expressive power of MLPs, severe overfitting was observed. Although Model D incorporated regularization techniques such as data augmentation, it still failed to capture the inductive biases inherent to CNNs or properly learn the facial features required for the task. Consequently, the test accuracy of the Baseline remained at 56.4%, and even Model D—with a deeper MLP—achieved only 61.3%. Model D showed some improvement, but its performance still remained insufficient for the task.

Model A - C

Models A - C reused the pretrained FC head from the Baseline while progressively unfreezing earlier convolutional layers for fine-tuning. In Model A, fine-tuning only the last conv layer, thus adapting only high-level features, already led to substantial improvement. Models B and C, which fine-tuned increasingly coarse feature extractors, demonstrated even stronger generalization performance. Based on the Size–Similarity Matrix, we predicted that the degree of fine-tuning between Models B and C would be the most appropriate for a Quadrant 3 scenario; indeed, Model C achieved the best results. This improvement can be attributed not only to the stronger data augmentation but also to the correction of a mismatch introduced earlier when the stem layer's stride was modified without retraining during the Baseline construction. Through full fine-tuning in Model C, this mismatch was naturally resolved, enabling the model to reach a test accuracy of **95.4%**, the highest among all models.

Data Augmentation

Additionally, data augmentation played a crucial role across Models A - D in mitigating overfitting and improving generalization. As more layers are fine-tuned, the effective dataset size becomes smaller relative to model capacity, making overfitting more likely. However, by exploiting prior knowledge such as the translation invariance of CNNs, the symmetry of human faces, and the color variability present in the training images, we were able to maximize the effectiveness of data augmentation and overcome the limitations of the small dataset. This suggests that, even with limited data, expanding the scope of fine-tuning can be highly effective when supported by strong and appropriately designed augmentation strategies.

Question 4

CAM-based Automatic Localization

The Fully Convolutional ResNet18 (FCResNet18) generates class-specific score maps by computing the weighted sum of feature maps from the final convolutional layer. For each predicted class, the network produces a spatial heatmap where high-activation regions indicate discriminative features. Bounding boxes are automatically extracted by thresholding these score maps and finding the minimum enclosing rectangle around the activated regions. This training-free approach leverages the spatial information preserved in fully convolutional architectures, enabling object localization without explicit bounding box annotations.

Summary of Key Findings

1. **Face Detection Comparison (Ryu vs. Cha):** The contrasting results between Ryu Hyun-jin (success) and Cha Eun-woo (failure) reveal a fundamental limitation: ImageNet-1k lacks direct human/face classes. Success depends on contextual objects—the baseball uniform and glove provided semantic anchors for 'Ballplayer', while a plain portrait offered no such cues. This demonstrates that CAM-based detectors perform zero-shot classification implicitly; targets outside the training taxonomy cannot be detected regardless of image quality.

2. **Multi-object Scene (Cat Crops):** The ablation study on the two-cat image exposed multiple failure modes. Background texture (bokeh blur) dominated the classification, and even after cropping, the model struggled with instance separation. CAM methods inherently lack instance awareness—they highlight class-relevant regions but cannot distinguish individual objects of the same class.

3. **Texture Bias vs. Taxonomy Limitation:** While CNNs exhibit well-documented texture bias (as seen in the 'Shower Curtain' misclassification), the face detection failures stem primarily from taxonomy gaps. These represent distinct failure modes requiring different solutions: texture bias can be mitigated through augmentation or architectural changes (e.g., ViT), whereas taxonomy limitations require training data expansion or open-vocabulary detection methods.

Conclusion

In this project, we tackled a series of well-structured problems that followed a coherent progression based on the theories and approaches covered throughout the semester—from non-linear regression, to using CNN as feature extractors for image dimension reduction, to learning discriminative features for a face recognition dataset via transfer learning on ResNet, and finally applying a classification model to a different task such as detection.

Through the project, we were able to gain hands-on experience not only in applying theoretical approaches, but also in building the end-to-end pipeline using deep learning frameworks such as PyTorch. This included constructing datasets, implementing models, setting up training pipelines, and directly configuring various regularization and optimization strategies to improve generalization performance. In addition, by implementing visualizations such as curve fitting results, feature distributions of the CIFAR-10 dataset, and response maps of FCResNet18, we were able to move beyond treating large datasets and models as mere black boxes, and instead gain a much deeper level of understanding that cannot be obtained from theory and equations alone.

Furthermore, in previous assignments and quizzes, the data we used were either artificially generated for learning purposes or small, simple datasets. In contrast, this project provided a valuable opportunity to work directly with large-scale datasets, such as CIFAR-10 and face recognition dataset, including data preprocessing and augmentation. In doing so, we experienced firsthand that solving problems in non-ideal settings requires much more than simply feeding an ideally constructed large dataset into a well-designed high-capacity model. It demands careful consideration of many aspects, such as the importance and similarity of datasets, processes for hyperparameter selection like k-fold validation, model capacity and complexity, and optimization strategies, all grounded in solid theoretical foundations.

Lastly, we believe that the insights gained from the Introduction to AI course and this project will serve as a strong foundation for understanding more advanced topics that appear in computer vision and natural language processing research papers, as well as for conducting future research in artificial intelligence.

Contributions

- Introduction: Junsoo Ham(50%), Jin Chun(50%)
- Problems and Solution
  Question 1. Junsoo Ham(50%), Jin Chun(50%)
  Question 2. Junsoo Ham(50%), Jin Chun(50%)
  Question 3. Hyeonbeen Jeon(50%), John Yechan Jo(50%)
  Question 4. Hyeonbeen Jeon(50%), John Yechan Jo(50%)
- Discussion: Junsoo Ham(25%), Hyeonbeen Jeon(25%), Jin Chun(25%), John Yechan Jo(25%)
- Conclusion: Hyeonbeen Jeon(50%), John Yechan Jo(50%)