Introduction to Machine Learning: Exercise 4

Yonghao Lee

December 25, 2025

1 Introduction

This report details the implementation and optimization of a Multi-Layer Perceptron (MLP) for spatial classification and the evaluation of Convolutional Neural Networks (CNNs) for deepfake detection. We utilized a dataset consisting of European city coordinates (longitude and latitude) to classify them into their respective countries.

2 Evaluating MLPs Performance (Sec. 6.2)

In this section, we analyze the performance of various MLP architectures (varying depth and width) to understand their generalization capabilities.

2.1 Question 1: Best Model Analysis

After training all 7 classifiers, the best model was identified as **Depth 6, Width 16** (Val Acc: 0.9588).

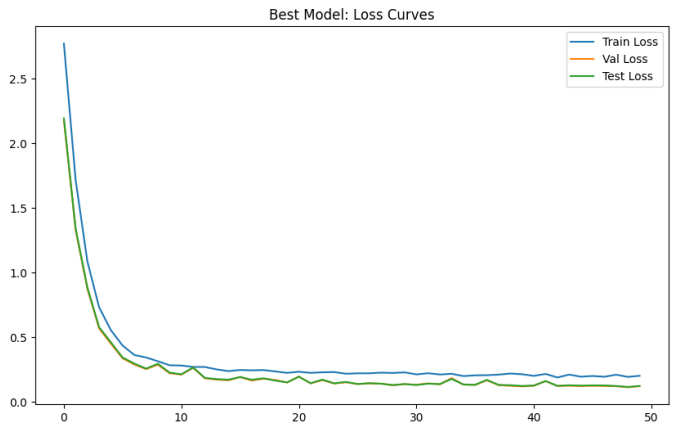


Figure 1: Best Model Loss Curves

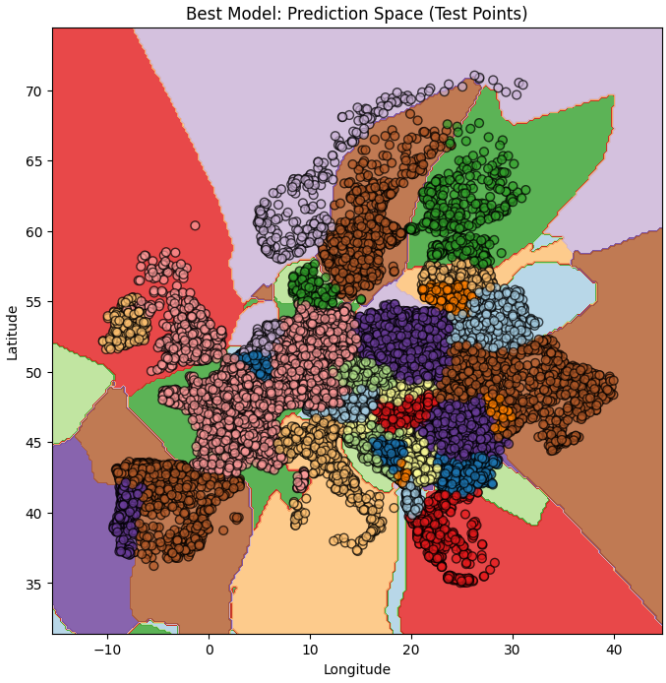


Figure 2: Best Model Prediction Space

Test Acc: $\approx 95.5\%$ — **Val Acc:** $\approx 95.9\%$
The model generalized exceptionally well. The test accuracy is very close to the validation accuracy, indicating that the model learned the underlying geographic boundaries effectively without overfitting.

2.2 Question 2: Worst Model Analysis

The worst model was **Depth 1, Width 16** (Val Acc: 0.8948).

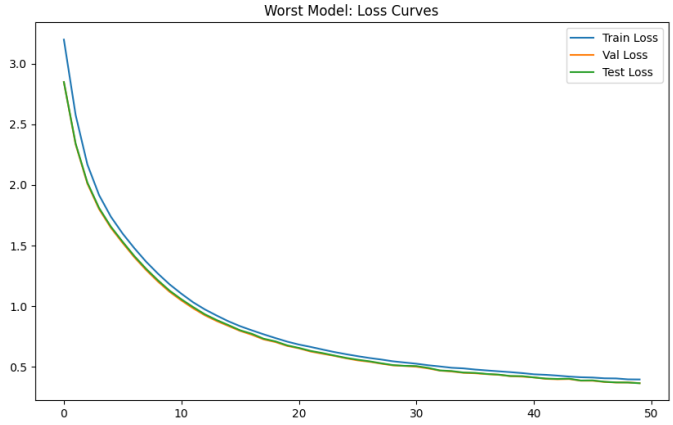


Figure 3: Worst Model Loss Curves

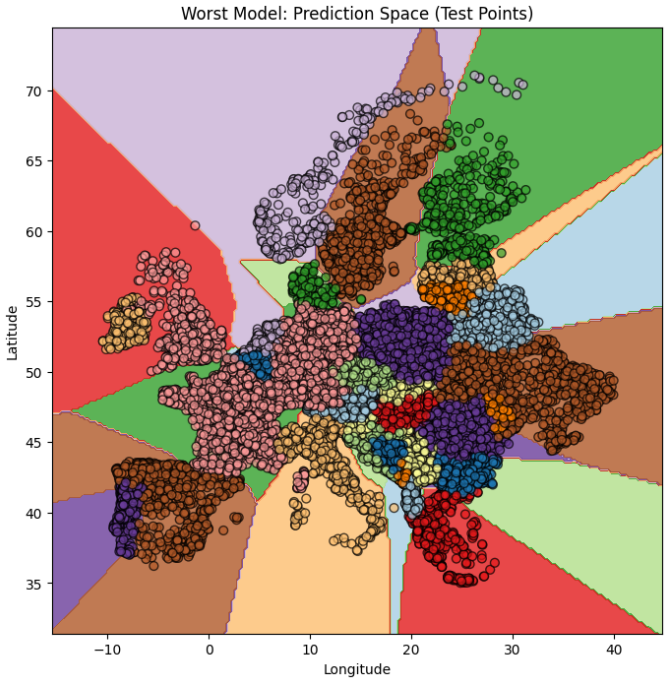


Figure 4: Worst Model Prediction Space

With only 1 hidden layer, the model is likely **underfitting**. It is too simple to capture complex geographical patterns, resulting in significantly lower accuracy than deeper or wider versions.

2.3 Question 3: Depth Analysis

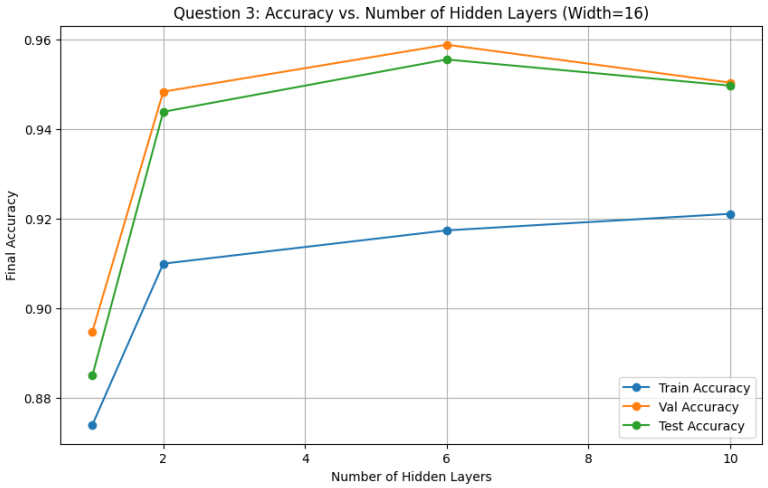


Figure 5: Accuracy vs. Number of Hidden Layers (Width=16)

Analysis:

- A significant leap in accuracy occurs between depth 1 and 2, suggesting single-layer networks lack capacity for non-linear geographical modeling.
- Peak performance is achieved at 6 layers. Beyond this, training accuracy continues to increase slightly while validation/test accuracies decline, indicating the onset of overfitting.

- Notably, training accuracy remains slightly lower than validation/test accuracy. This is likely due to **Batch Normalization** behaving differently: in `train()` mode, it introduces stochastic noise via batch-specific statistics, whereas in `eval()` mode, it utilizes stable running statistics.

2.4 Question 4: Width Analysis

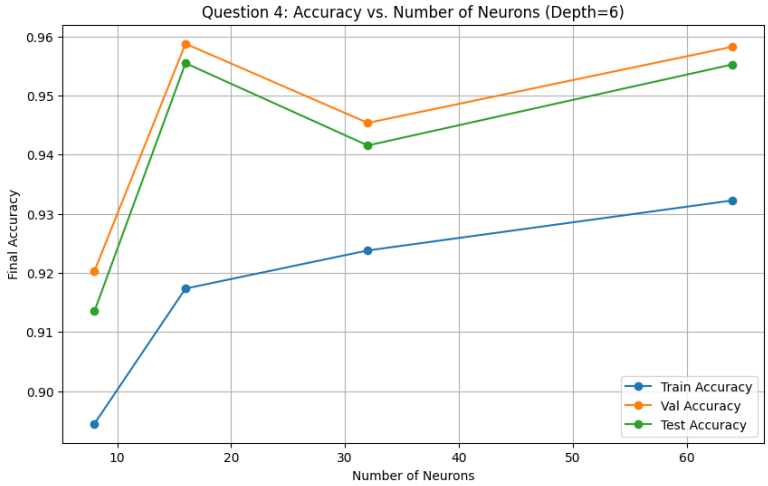


Figure 6: Accuracy vs. Number of Neurons (Depth=6)

Analysis:

- Performance increases sharply as additional neurons expand the model's capacity to learn features in parallel.
- The dip at width 32 likely indicates the model became stuck in a local minimum or a flat plateau. This is often a byproduct of high-stability training lacking the stochastic noise needed to escape sub-optimal regions.

2.5 Question 5: Gradient Dynamics

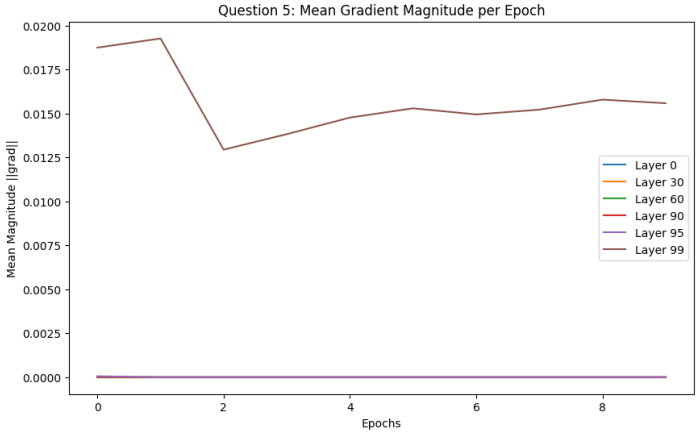


Figure 7: Mean Gradient Magnitude per Epoch (100 Layers)

Analysis: Monitoring a 100-layer network reveals a severe **vanishing gradient** phenomenon. Layer 99 remains the only healthy layer; as the signal propagates backward, magnitudes for Layers 0–95 decay toward 0.0. This occurs because the gradient is repeatedly multiplied by small weights and activation derivatives, causing the signal to decay exponentially. This can be solved by using a **residual network** to create pathways that skip backward.

2.6 Question 6: Residual MLP

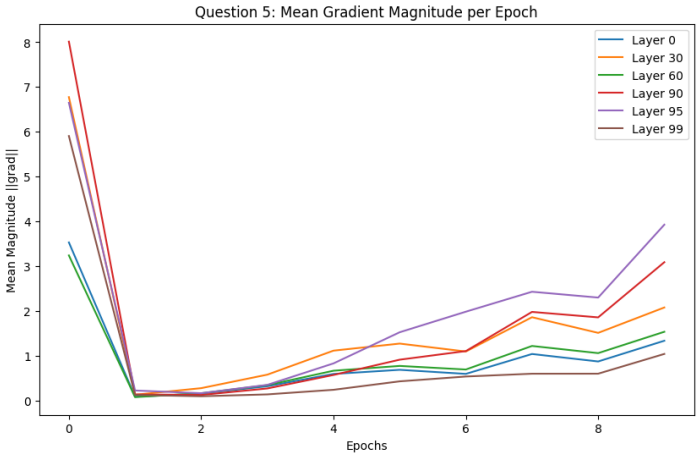
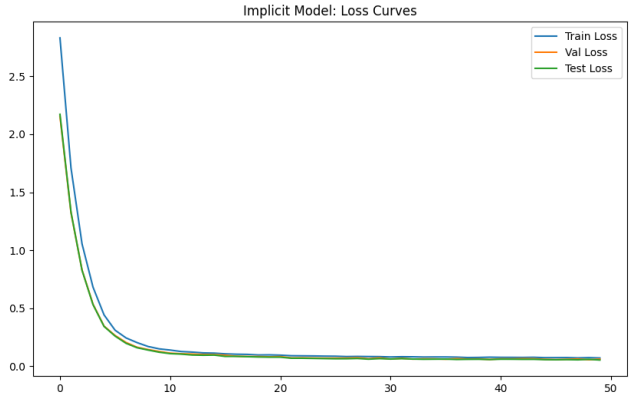


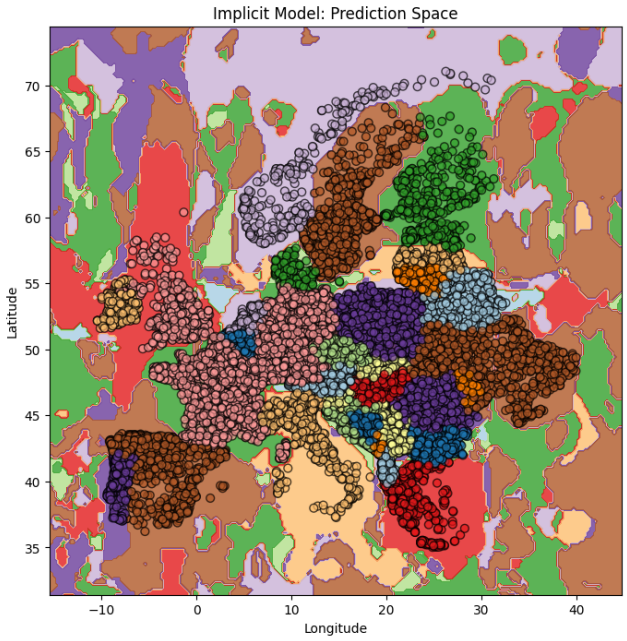
Figure 8: Residual MLP Magnitude Comparison

Analysis: The Residual MLP solved the vanishing gradient problem. Even Layer 0, the furthest from the output, is learning as effectively as the rest. After initial spikes, the gradients of all layers converge in magnitude, showing that the network is learning as a cohesive unit.

2.7 Question 7: Implicit Representation (Bonus)



(a) Implicit Model: Loss Curves



(b) Implicit Model: Prediction Space

Figure 9: Implicit Representation Model Performance

Analysis: The accuracy increased by about 2% compared to the standard Depth 6/Width 16 architecture. By projecting 2D input into a 20-D space, the network distinguishes coordinates that are numerically close but belong to different categories. The sine transformation provides high-frequency information, producing more detailed ("wiggly") borders. This pre-processing allows the network to focus on complex patterns rather than basic periodic functions.

3 Optimization of an MLP (Sec. 6.1)

The base model architecture used for optimization is a 6-layer MLP (including the input layer but not the output layer, totaling 7 linear instances) with ReLU activations.

3.1 Question 1: Learning Rate Analysis

We trained the network using four different learning rates: $1.0, 0.01, 0.001$, and 0.00001.

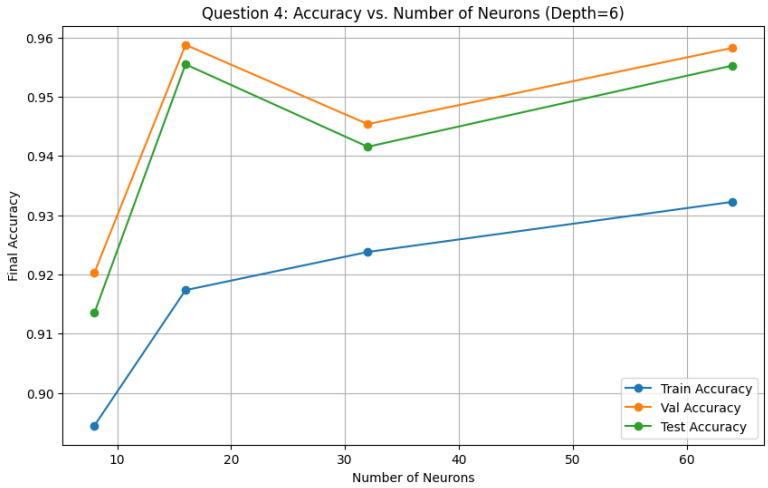


Figure 10: Validation Loss for Different Learning Rates over 50 Epochs

Learning Rate Analysis: As shown in Figure 10, the loss curves exhibit noticeable fluctuations and spikes. This behavior is expected because we have not yet introduced **Batch Normalization**. Without it, the model is highly sensitive to the **internal covariate shift**—where the distribution of inputs to each layer changes during training—and to the specific variance of each mini-batch.

- **Too High ($LR = 1.0$):** The loss remains stagnant at a high level. This occurs because the optimizer takes steps so large that it aggressively overshoots the local minima, preventing the model from ever descending the loss landscape toward convergence.
- **Too Low ($LR = 0.00001$):** A low learning rate results in extremely slow convergence. While the descent is steady and smooth, the steps toward the minimum are too minute to reach an optimal solution within 50 epochs.
- **Optimal ($LR = 0.01$ or 0.001):** These rates provide the ideal balance, allowing for efficient and relatively stable convergence. The occasional spikes in the loss are characteristic of raw MLP training where the gradients of specific batches may temporarily push the weights into less stable regions, a phenomenon typically mitigated by the smoothing effect of Batch Normalization layers.

3.2 Question 2: Epoch Analysis

Following the optimization of the learning rate, we selected $LR = 0.001$ to evaluate the impact of training duration over 100 epochs.

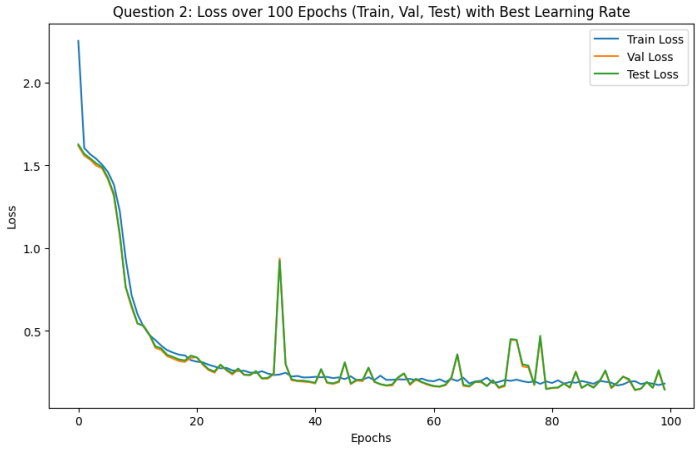


Figure 11: Loss over 100 epochs for Training, Validation, and Test sets ($LR = 0.001$).

Analysis: The relationship between the number of epochs and model performance can be categorized into three stages:

- **Too Few Epochs (Underfitting):** In the early stages of training (approximately epochs 0–20), all three loss curves drop rapidly. Stopping the training in this phase results in underfitting, as the model has not yet had sufficient iterations to capture the underlying geographical patterns in the dataset.
- **Optimal Epochs:** Between epochs 40 and 60, the curves reach their lowest stable points. At this stage, the model has achieved a balance where it has learned the data patterns effectively without yet over-specializing on the training set noise.
- **Too Many Epochs (Overfitting):** As training progresses toward 100 epochs, we observe a divergence in behavior. While the Training Loss becomes extremely steady and continues a slow, flat decline, the Validation and Test losses exhibit increased fluctuations and spikes. This indicates overfitting, where the model begins to memorize specific noise or outliers in the training data, leading to a decreased ability to generalize to unseen coordinates.

3.3 Question 3: Batch Norm Analysis

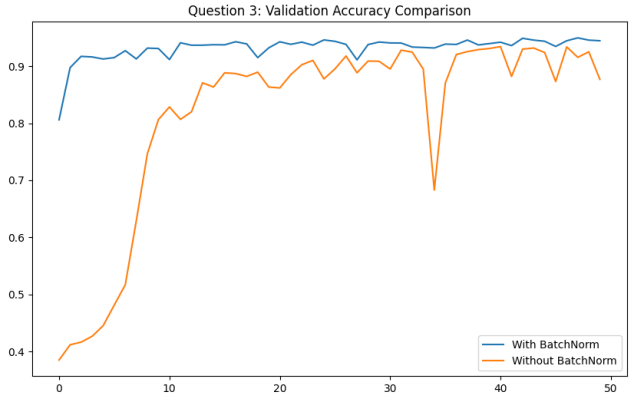


Figure 12: Validation accuracy comparison with and without Batch Normalization ($LR = 0.001$).

Analysis: The introduction of Batch Normalization layers significantly improved both the speed of convergence and the stability of our spatial classification model.

- **Faster Convergence:** Within the first 5 epochs, the accuracy of training with batch norm achieves above 90%, whereas the version without takes significantly more epochs to achieve comparatively mediocre accuracy.

- **Smooth Loss Changes:** The training without batch norm suffers a significant drop in accuracy at around $epoch = 33$, whereas batch normalization prevents this by normalizing the distribution of inputs to the following layers.
- **Superior Final Results:** Clearly, through the run of all allocated epochs, the performance of training with batch norm is superior to its unnormalized version.

3.4 Question 4: Batch Size Comparison

We evaluated the performance using batch sizes of $1, 16, 128$, and 1024.

Analysis:

1. **Accuracy vs. Epochs**:

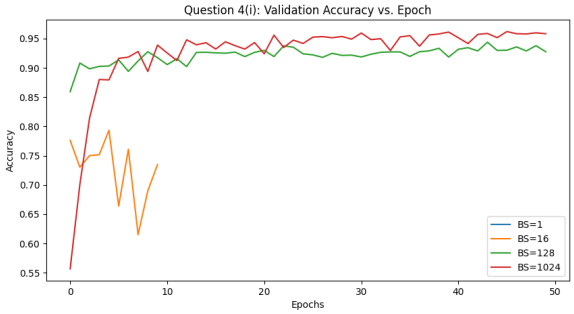


Figure 13: Validation Accuracy vs. Epoch

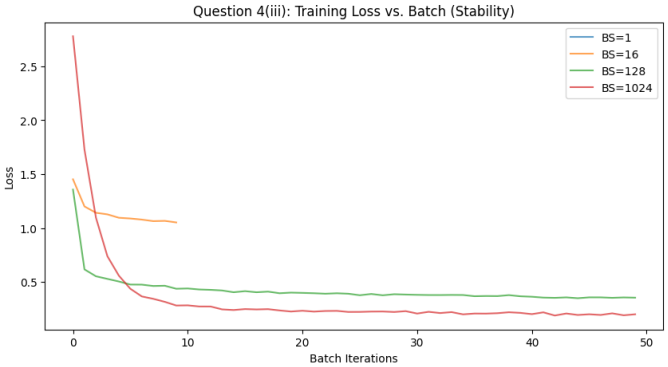


Figure 14: Training Stability (Loss vs. Batch)

The $BS = 1024$ configuration achieved the highest final validation accuracy of approximately 95.8%. While $BS = 1$ achieved a surprisingly high accuracy of 56.7% in just a single epoch, it lacked the refinement seen in larger batch runs. The $BS = 16$ curve exhibited the most instability, with accuracy fluctuating significantly between epochs (e.g., dropping from 79% to 66%), indicating that smaller batches introduce high stochastic noise that can hinder stable convergence.

2. **Speed and Iterations**: Iterations per epoch are inversely proportional to the batch size:
   - Batch Size 1: 106897 iterations per epoch
   - Batch Size 16: 6681 iterations per epoch
   - Batch Size 128: 835 iterations per epoch
   - Batch Size 1024: 104 iterations per epoch
3. **Stability**: The $BS = 1024$ line is the most stable and smooth. This is because the gradient calculated at each step is an average of 1024 samples, providing a much more accurate estimate of the true gradient compared to $BS = 16$ or $BS = 1$, where individual outliers can pull the weights in suboptimal directions.

4 Convolutional Neural Networks for Deepfake Detection (Sec. 7)

In this section, we evaluate several strategies for detecting deepfake images using the ResNet18 architecture and traditional machine learning baselines. We focus on comparing training from scratch, transfer learning (linear probing and fine-tuning), and non-CNN approaches.

4.1 Baseline Performance Comparison

We conducted an extensive evaluation of five distinct baselines. For the PyTorch-based models, we performed a hyperparameter sweep across five learning rates (10^{-1} to 10^{-5}) and trained for exactly one epoch. The results are summarized in Table 1.

Baseline Strategy	Best Learning Rate	Test Accuracy
1. XGBoost (Raw Pixels)	N/A	0.7350
2. ResNet18 (From Scratch)	0.01	0.5750
3. ResNet18 (Linear Probing)	0.01	0.7325
4. Sklearn Linear Probing (Bonus)	N/A	0.6900
5. ResNet18 (Fine-tuning)	0.0001	**0.8500**

Table 1: Comparison of Deepfake Detection Baselines after 1 Epoch.

Analysis:

- **Best Model:** The **Fine-tuning** strategy achieved the highest accuracy of 0.8500. By allowing all layers of the pre-trained ResNet18 to adapt, the model successfully leveraged high-level features from ImageNet while refining them for the subtle artifacts found in deepfakes.
- **Second Best Model:** The **XGBoost** model (0.7350) was the second-best performer, marginally outperforming the Linear Probing strategy (0.7325). This highlights the effectiveness of gradient boosting on raw pixel data when deep learning training time is limited to a single epoch.
- **Worst Model: Training from Scratch** performed the worst (0.5750). Within a single epoch, the complex ResNet18 architecture cannot learn effective facial feature extraction from random initialization, highlighting the necessity of pre-training for small datasets or limited training windows.
- **Learning Rate Trends:** Fine-tuning required a significantly lower learning rate (10^{-4}) compared to Linear Probing (10^{-2}). This is standard for transfer learning; a high learning rate in fine-tuning can damage the fragile, pre-trained weights.

4.2 Qualitative Error Analysis

To understand the models' decision-making processes, we compared the predictions of our best model (Fine-tuning, $LR = 10^{-4}$) and our worst model (Scratch, $LR = 10^{-1}$).

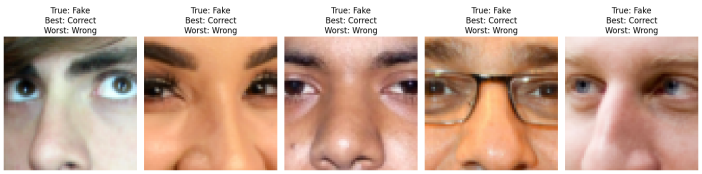


Figure 15: Samples where Fine-tuning correctly identified deepfakes that the Scratch model missed.

Analysis: The samples in Figure 15 highlight the superior feature extraction capabilities of the fine-tuned ResNet18. The "From Scratch" model, lacking sufficient training time to develop anatomical priors, failed to identify these images as fake. In contrast, the Fine-tuning model correctly identified subtle GAN artifacts, such as:

- **Anatomical Inconsistencies:** Irregular pupil shapes and asymmetrical eye positioning.
- **Structural Artifacts:** Distortions around complex objects like glasses and the bridge of the nose where generated textures often appear blurred or digitally "noisy".

This demonstrates that transfer learning allows the model to utilize pre-existing knowledge of human faces to detect the minute deviations present in generated imagery.