Mask Detection using CNNs (Part B)

1 Introduction

This project aims to classify images of faces as **"with mask"** or **"without mask"** using **Convolutional Neural Networks (CNNs)**. In **Part A**, we used **feature extraction with multiple machine learning models** (SVM, MLP, XGBoost, etc.), whereas in **Part B**, we focus on **deep learning-based classification** using CNNs.

We divide Part B into two phases:

- **Normal Analysis**: Uses smaller CNN architectures with different activation functions and optimizers.
- **Advanced Analysis**: Uses larger CNN models (including ResNet-like and MobileNet).

The goal is to compare the effectiveness of simple vs. advanced CNN architectures for mask detection.

2 Dataset & Preprocessing

The dataset is preprocessed into **NumPy arrays** (`X.npy`, `y.npy`) for efficient loading and training.

- **Image Size**: $128 \times 128 \times 3$
- **Classes**: With mask (1) and without mask (0)
- **Train-Test Split**: 80%-20%

3 Normal Analysis

The **Normal Analysis** focuses on simple CNN models trained directly in **Jupyter Notebook**.

3.1 Model Architecture

Three models were trained with varying activation functions and optimizers:

- **Model 1**: ReLU + Adam
- **Model 2**: Tanh + Adam
- **Model 3**: ReLU + SGD

Common Architecture:

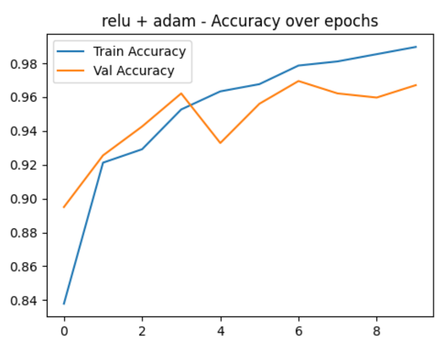
- Conv2D(32) → MaxPooling
- Conv2D(64) → MaxPooling
- Conv2D(128) → MaxPooling → Flatten → Dense(128) → Output Layer

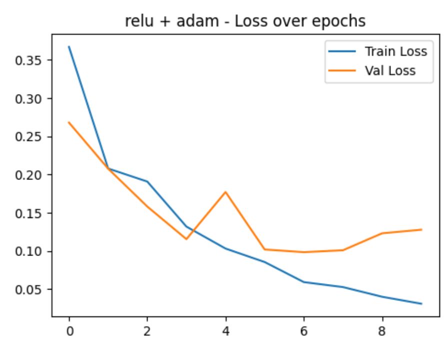
3.2 Training Process

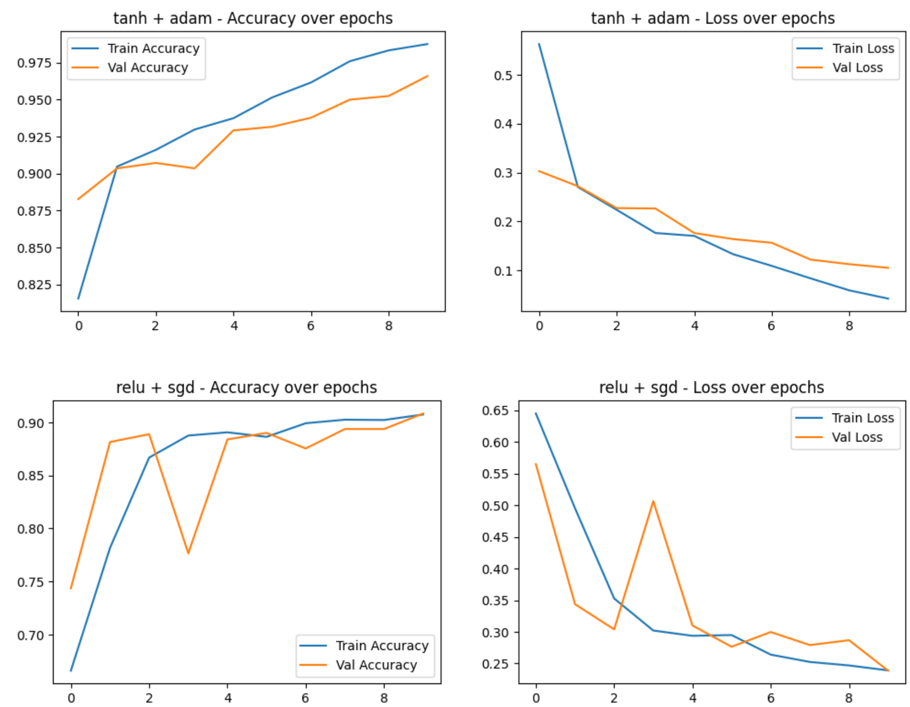
- **Epochs**: 10
- **Batch Size**: 32
- **Loss Function**: Binary Crossentropy
- **Evaluation Metric**: Accuracy

3.3 Evaluation

Model	Validation Accuracy
ReLU + Adam	96.70%
Tanh + Adam	96.58%
ReLU + SGD	90.84%







4 Advanced Analysis

4.1 Motivation for Advanced Analysis

Although **Normal Analysis** achieved high accuracy, its models were simple and might not generalize well. To improve, we explore **larger architectures** using **Google Colab (GPU acceleration)**.

4.2 Model Architectures

- **Baseline CNN** – Similar to Normal Analysis but deeper.
- **VGG-like CNN** – Inspired by VGG architecture, uses more layers.
- **ResNet-like CNN** – Introduces residual connections.
- **MobileNet** – Lightweight, designed for mobile applications.

4.3 Training Process

- **Epochs**: 25
- **Batch Size**: 64
- **Optimizer**: Adam

4.4 Evaluation

Model	Validation Accuracy	Validation Loss
Baseline CNN	97.68%	0.1619
VGG-like CNN	96.09%	0.2767
ResNet-like CNN	95.48%	0.2748
MobileNet	43.83%	2.3555

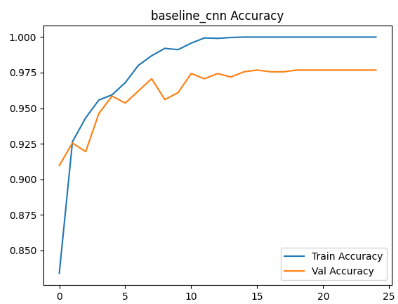


Figure 1: Baseline CNN

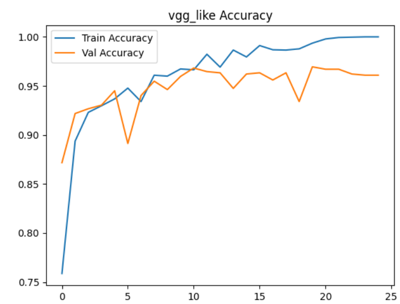


Figure 2: VGG-like Model

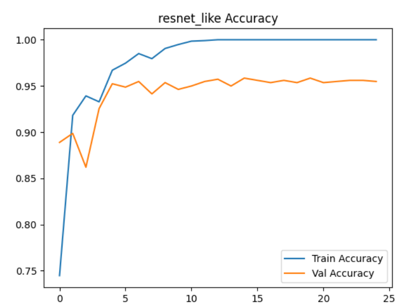


Figure 3: ResNet-like Model

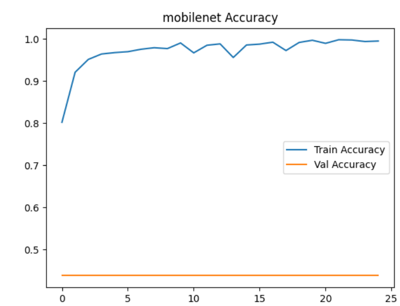


Figure 4: MobileNet Model

5 Differences Between Models

5.1 Model Complexity & Architecture

Model	Key Features	Intended Benefit
Baseline CNN	Simple, 3 Conv layers	Fast training
VGG-like CNN	5 Conv layers	Strong feature learning
ResNet-like CNN	Residual connections	Improved gradient flow
MobileNet	Depthwise separable convolutions	Optimized for mobile

5.2 Computational Efficiency

Model	Parameters	Training Time
Baseline CNN	∼1.2M	Fast
VGG-like CNN	∼3.5M	Slower
ResNet-like CNN	∼2.9M	Moderate
MobileNet	∼2.2M	Slow (unexpected)

5.3 Performance & Accuracy

Model	Validation Accuracy
Baseline CNN	97.68%
VGG-like CNN	96.09%
ResNet-like CNN	95.48%
MobileNet	43.83%

6 Model Performance Visualization

6.1 Comparative Observations

Model Type	Best Performing Model	Overfitting Risk
Normal Analysis	ReLU + Adam	Low
Advanced Analysis	Baseline CNN	Low

6.2 Conclusion: More complex doesn't always mean better

- **Baseline CNN outperformed deeper models**.
- **Simpler models in Normal Analysis performed nearly as well as Advanced Analysis models**.
- **MobileNet failed**, reinforcing the importance of pretrained weights.