# IMAGE CLASSIFICATION WITH DEEP LEARNING

From Simple CNNs to Advanced Transfer Learning

Julia, Maryem and Helena

CIFAR-10 IMAGE CLASSIFICATION PROJECT

-

# Project Overview

- ✓ **Goal:** Build an accurate image classification model for CIFAR-10 dataset
- ✓ **Dataset**: 60,000 images across 10 classes (32×32 RGB)
- ✓ **Challenge**: Improve accuracy through architecture design and optimization
- ✓ **Approach**: Systematic progression from simple to complex models

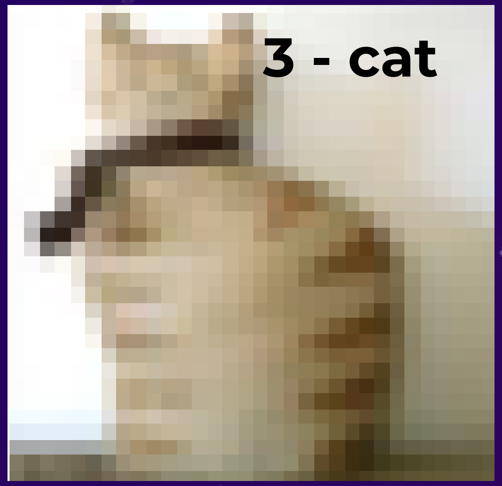
# CIFAR-10 dataset

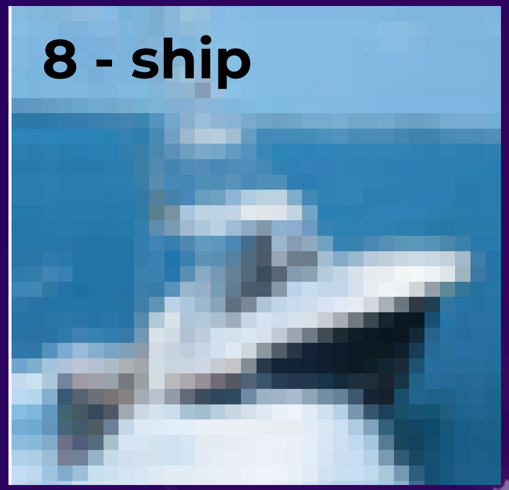
- **Total:** 60,000 images
- **Training dataset**: 50,000 images
- **Test dataset:** 10,000 images
- **Image dimensions:**
  - 32x32 pixels,
  - 3 channels (RGB)
- **10 classes:** 0-9

- Even distribution of images per each class:
  - 5,000 images per class

**PREPROCESSING:**

- Data was normalized:
  - *X_train = X_train/255*
  - *X_test = X_test/255*





# Our Approach

1. **Baseline CNN (5 layers)**
   Tune Hyperparameters

2. **VGG Style Architecture (10 layers)**
   Deeper Feature Extraction

3. **ResNet-20 (20 layers)**
   Skip connections

4. **Transfer Learning**
   - InceptionV3 -DenseNetV2 -EfficientNetV2


# CNN scratch

# Our VGG-10 CNN: Deeper with Optimizations

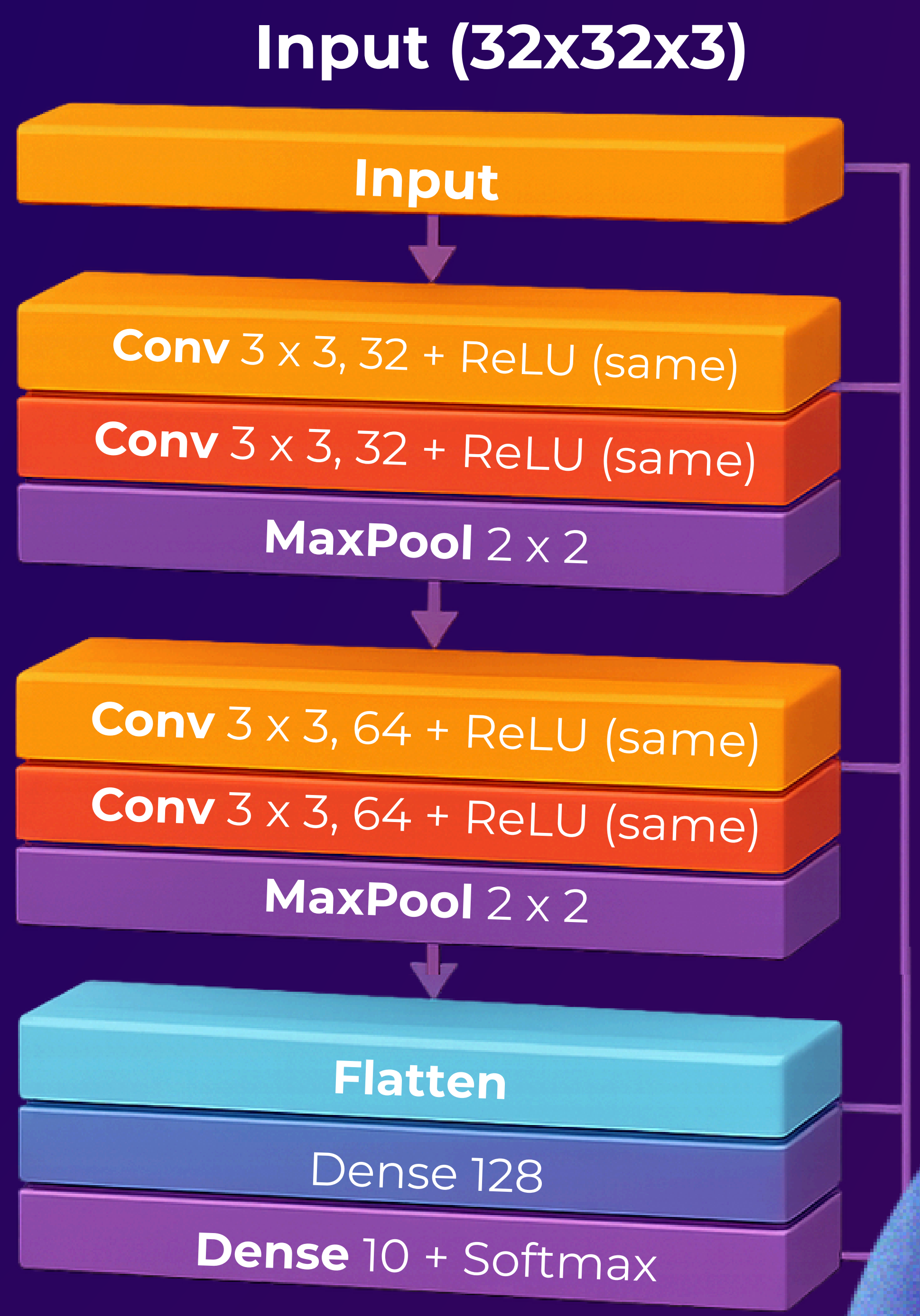
## VGG - (Visual Geometry Group, Oxford, 2014)

**Key idea**: *Go deeper using a simple, repeatable design*

- **Small filters:** uses 3×3 convolution filters
- **Block design:** Conv → Conv → MaxPool, repeat
- **Simple & consistent:** same building blocks throughout → easy to implement, and modify.
- **Practical impact:** became a popular ***baseline and feature extractor*** for many vision tasks

## Our tweaks:

- v Callbacks (Early stop, ReduceLROnPlateau)
- v Data augmentation
- v Dropout




**Baseline - VGG10**
**78.6%** |Loss: 0.64
20 epochs

**Callbacks**
**80.5%** |Loss: 0.64
20 epochs

**v Optimized**
**82.0%** |Loss: 0.54
50 epochs

# Our VGG-10 CNN: Deeper with Optimizations

## **VGG** - (Visual Geometry Group, Oxford, 2014)

**Key idea**: *Go deeper using a simple, repeatable design*

- **Small filters:** uses 3×3 convolution filters
- **Block design:** Conv → Conv → MaxPool, repeat
- **Simple & consistent:** same building blocks throughout → easy to implement, and modify.
- **Practical impact:** became a popular ***baseline and feature extractor*** for many vision tasks

## Our tweaks:

- v Callbacks (Early stop, ReduceLRonPlateau)
- v Data augmentation
- v Dropout

Input (32x32x3)

1. **Input**
2. **Conv** 3 x 3, 32 + ReLU (same)
3. **Conv** 3 x 3, 32 + ReLU (same)
4. **MaxPool** 2 x 2
5. **Conv** 3 x 3, 64 + ReLU (same)
6. **Conv** 3 x 3, 64 + ReLU (same)
7. **MaxPool** 2 x 2
8. **Flatten**/ Dense, 128
9. **Dense** 10 + Softmax

| **Baseline - VGG10** | **Callbacks** | **v Optimized** |
| --- | --- | --- |
| **78.6%** \|Loss: 0.64 | **80.5%** \|Loss: 0.64 | **82.0%** \|Loss: 0.54 |
| 20 epochs | 20 epochs | 50 epochs |

# ResNet-20 model

# CNN -2  deep model

# Evaluation and comparison

AI is integrated into various aspects of life, from smart home devices and chatbots to personalized recommendations on streaming services. In healthcare, AI aids in diagnostics, while in finance, it powers fraud detection and algorithmic trading.

AI is integrated into various aspects of life, from smart home devices and chatbots to personalized recommendations on streaming services. In healthcare, AI aids in diagnostics, while in finance, it powers fraud detection and algorithmic trading.

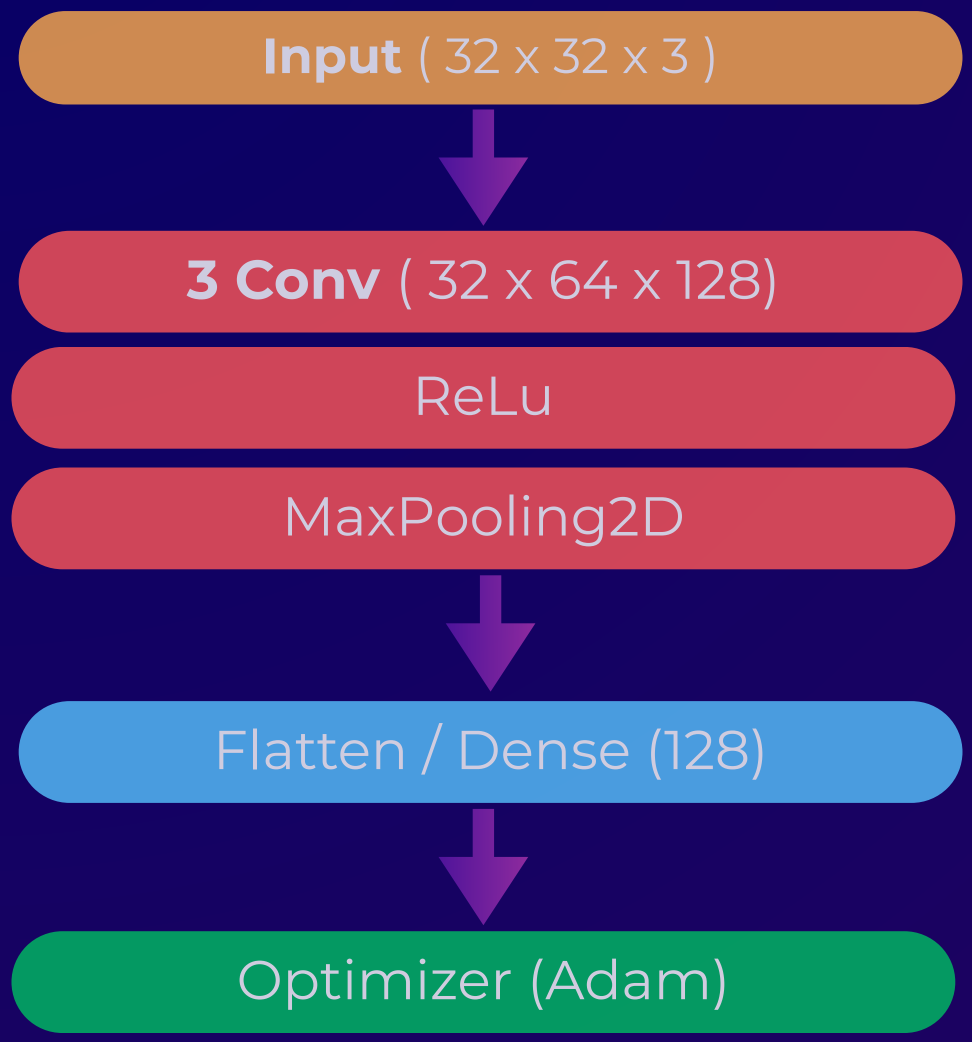
# CNN

# CNN scratch

# Extra - Preprocessing

- **Converting Numpy arrays to Tensorflow dataset**
- **Shuffle :** Randomly mixes training samples every epoch ( only training split )
- **Batch :** Groups individual samples into mini-batches, instead of loading everything at once
- **Prefetch :** Prepares the next batch while the model is training on the current one

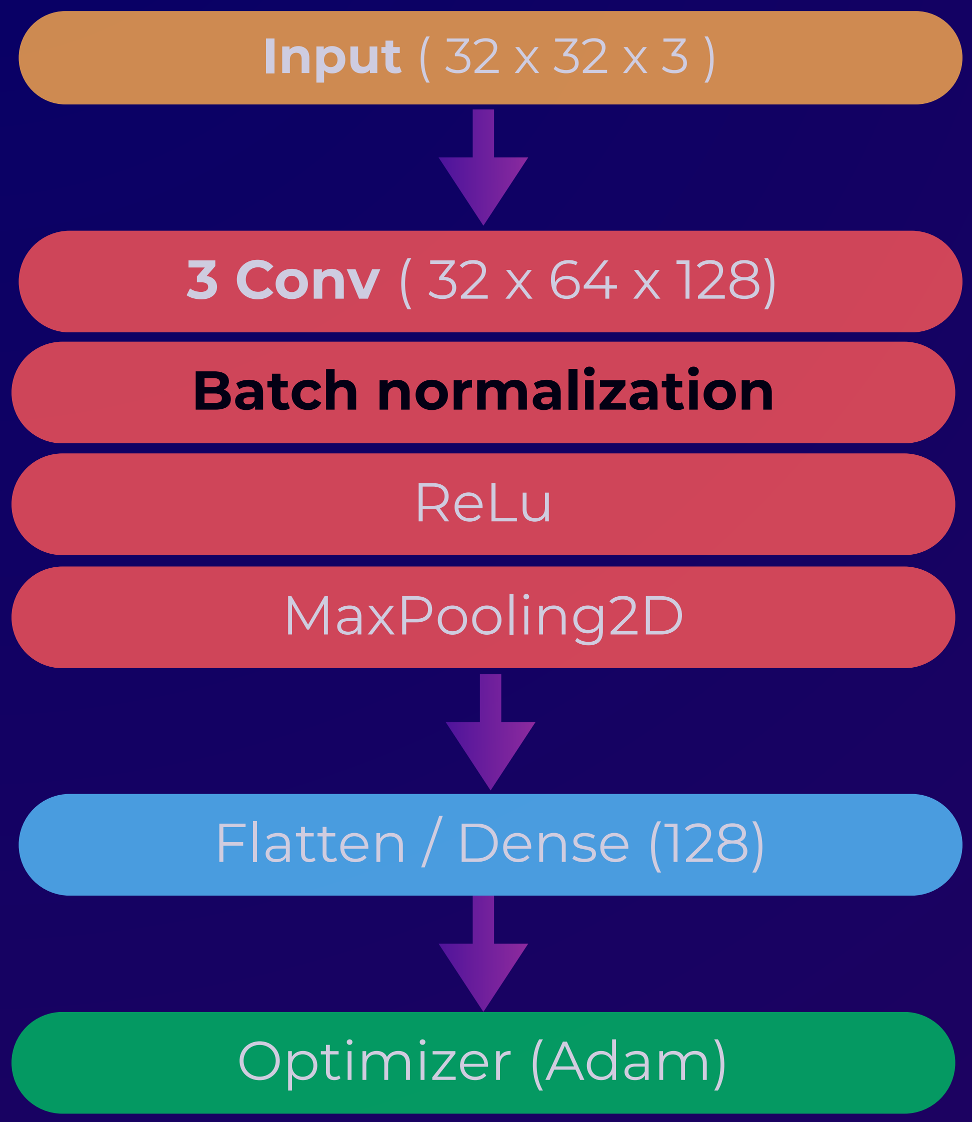
# Base Model




| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |

- Performance gap between train and test is indicating an overfitting
- Low test and validation , the model can't generalize effectively

# Model with batch normalization




| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |

- Instead of using mean and variance of the hole dataset we use normalized mini batch outputs (mean ≈ 0, variance ≈ 1), then applies learned scaling and shifting
- Add slight noise to ReLu → reduce overfitting.

- Faster and more stable training
- generalization has improved
- overfitting still present

# Model with data augmentation

**Input** ( 32 x 32 x 3 )

**3 Conv** ( 32 x 64 x 128)

Batch normalization

ReLu

MaxPooling2D

Flatten / Dense (128)

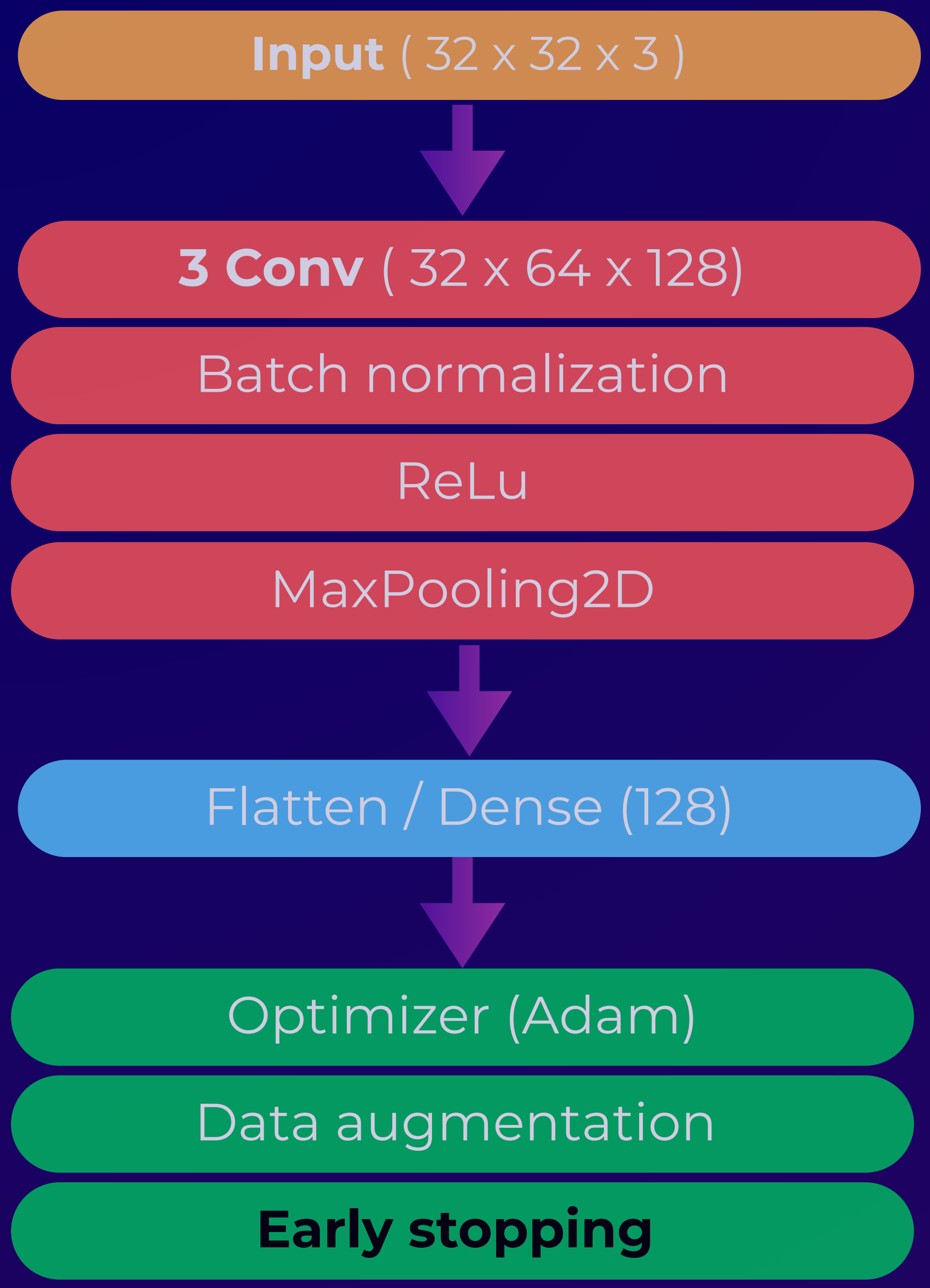
Optimizer (Adam)

**Data augmentation**

| **Model** | **Train Acc** | **Val Acc** | **Test Acc** |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |

- Data augmentation successfully reduced overfitting
- Test accuracy is still low but we observed that it peaked at epoch 17 and then dropped

# Model with early stopping




| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |

- The model stopped at epoch 18, preventing further overfitting.
- Early stopping allowed the model to capture useful patterns without overtraining
- Further reducing overfitting and improving generalization

# Model with dropout

**Input** ( 32 x 32 x 3 )

**3 Conv** ( 32 x 64 x 128)
Batch normalization
ReLu
MaxPooling2D

Flatten / Dense (128)
**Dropout (0,3)**

Optimizer (Adam)
Data augmentation
Early stopping

| **Model** | **Train Acc** | **Val Acc** | **Test Acc** |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |
| + Dropout | 74% | 66% | 71% |

- The dropout reduced overfitting aggressively, but in this case it caused underfitting
- Lower generalization capacity
- we will keep this change in order to prevent overfitting in the next experiments

# Model with global average pooling 2D

**Input** ( 32 x 32 x 3 )

↓

**3 Conv** ( 32 x 64 x 128)
Batch normalization
ReLu
MaxPooling2D

↓

**GlobAP** / Dense (128)
Dropout (0,3)

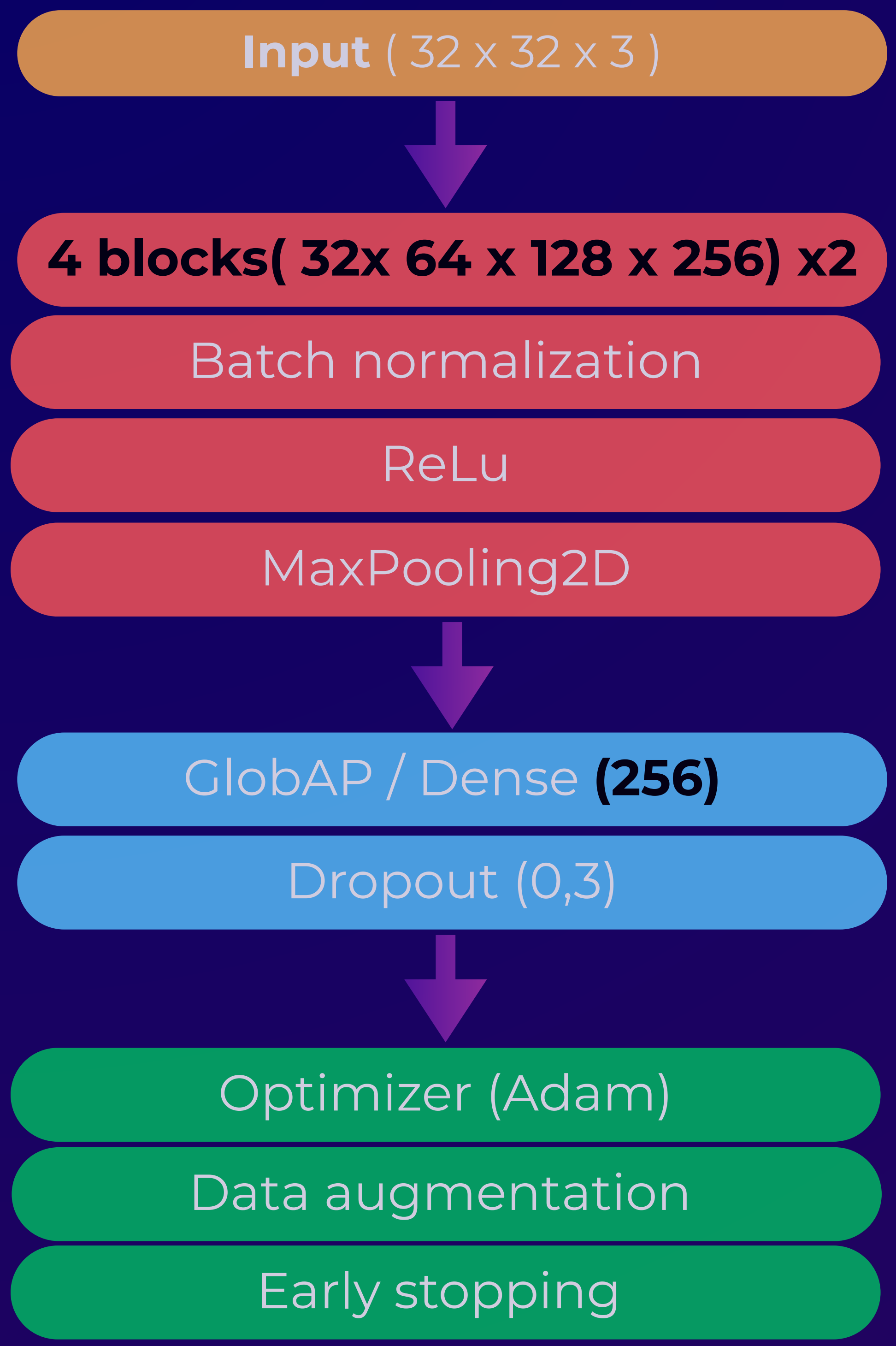
↓

Optimizer (Adam)
Data augmentation
Early stopping

| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |
| + Dropout | 74% | 66% | 71% |
| + GlobalaveragePool | 73% | 63% | 72% |

- Flattening increases the number of trainable parameters, which can lead to overfitting.
- Global Average Pooling (GAP) summarizes each feature map into a single value
- Lower generalization capacity
- Underfitting still present
- GAP may be more suitable for larger datasets or deeper models

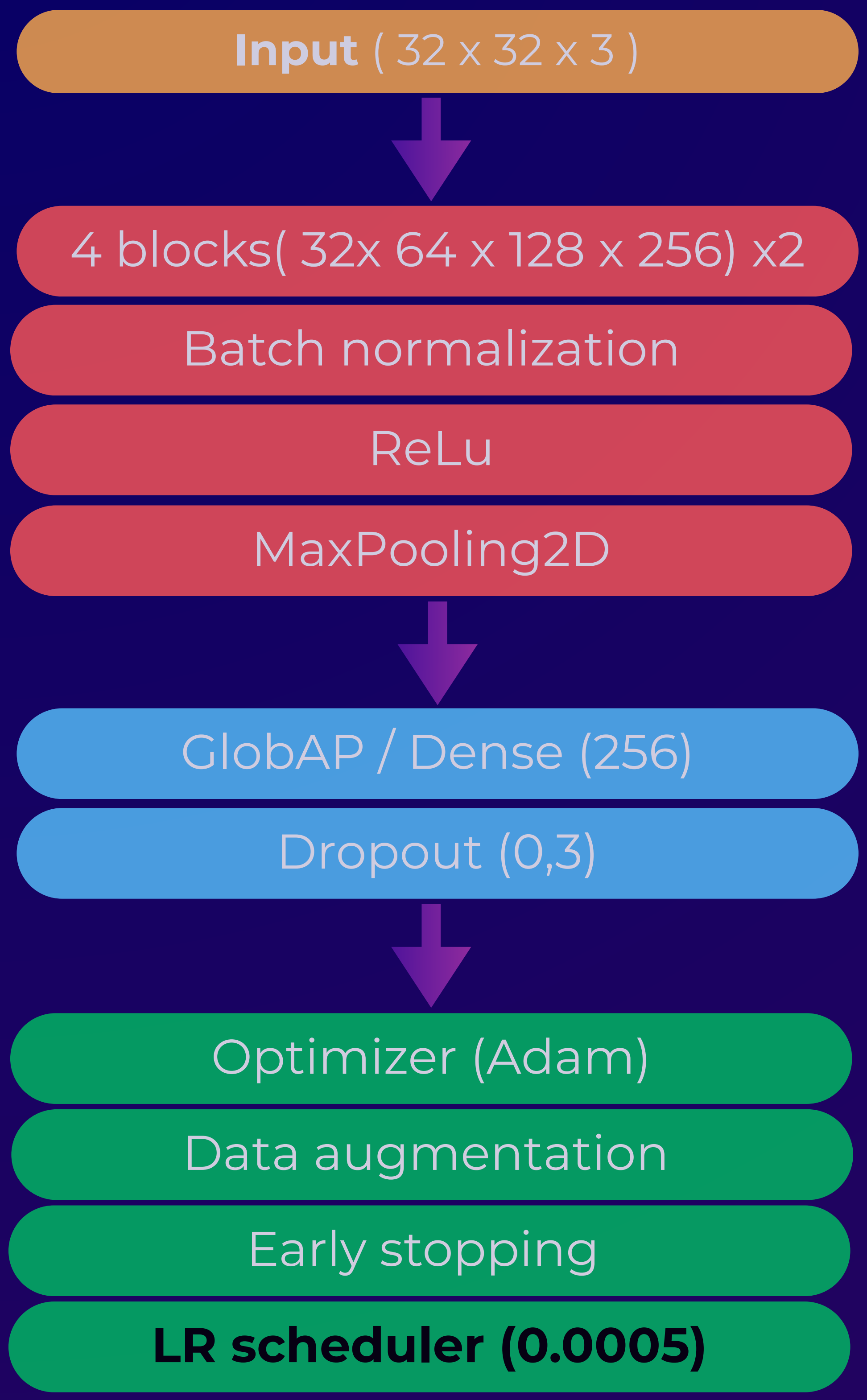
# Deeper model : 4 blocks




| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |
| + Dropout | 74% | 66% | 71% |
| + GlobalaveragePool | 73% | 63% | 72% |
| + Deeper model | 90% | 86% | 86% |

- Best generalization capacity so far .
- Moderate gap between training and test

# Model with LR scheduler




| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |
| + Dropout | 74% | 66% | 71% |
| + GlobalaveragePool | 73% | 63% | 72% |
| + Deeper model | 90% | 86% | 86% |
| + LR Scheduler | 92% | 88% | 88% |

- A learning rate scheduler (`ReduceLROnPlateau`) was added to dynamically adjust the optimizer’s learning rate during training
- While the 4-block model achieved strong performance, training with a fixed learningrate prevented the model from fully converging.
- Better feature refinement without increasing overfitting

# Model with AdamW

**Input** ( 32 x 32 x 3 )

4 blocks( 32x 64 x 128 x 256) x2
Batch normalization
ReLu
MaxPooling2D

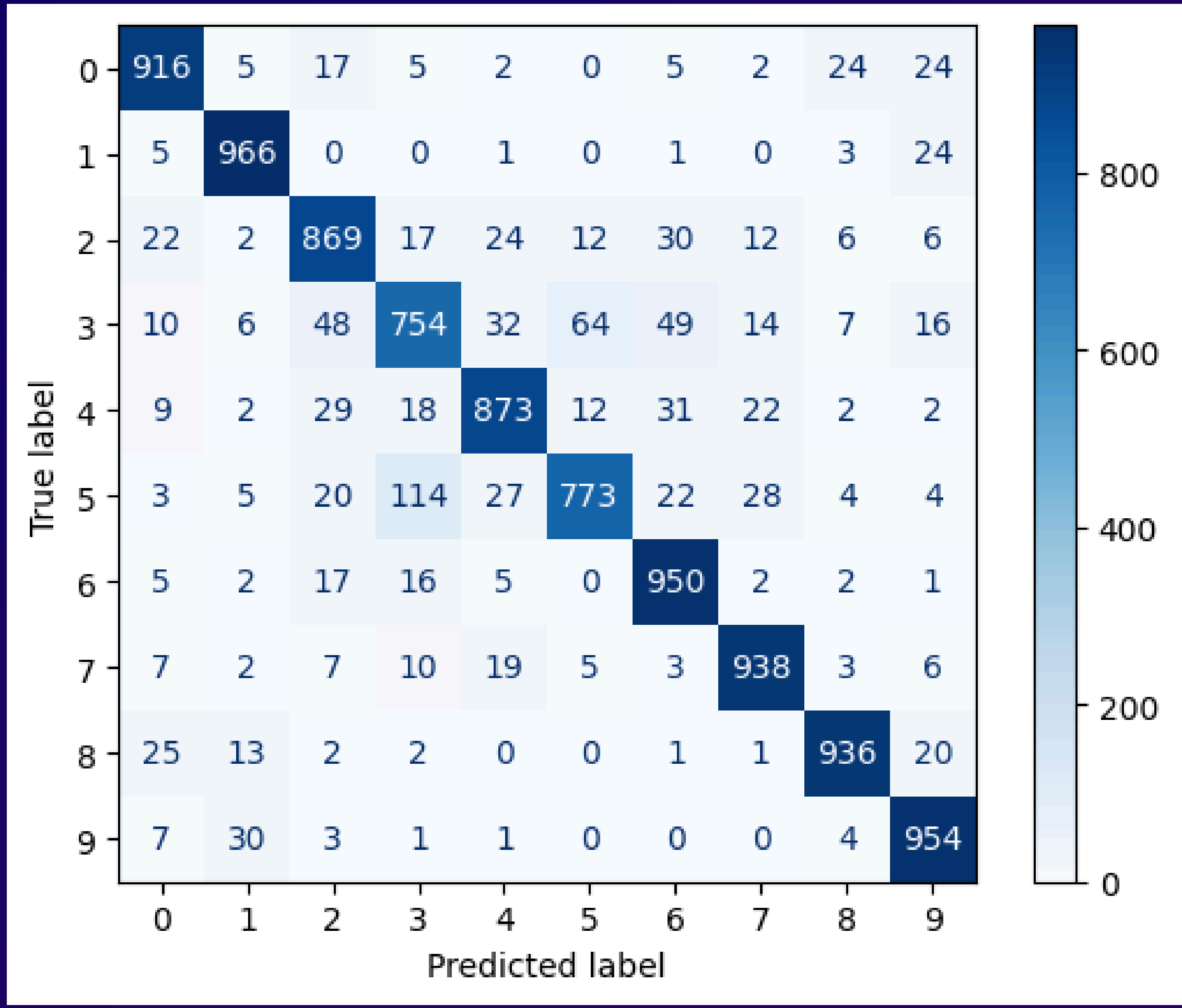
GlobAP / Dense (256)
Dropout (0,3)

**Optimizer (AdamW)**
Data augmentation
Early stopping
LR scheduler (0.0005)

| Model | Train Acc | Val Acc | Test Acc |
|---|---|---|---|
| Base | 93% | 73% | 73% |
| + Batch normalization | 98% | 76% | 75% |
| + Data augmentation | 83% | 78% | 78% |
| + Early stopping | 87% | 79% | 82% |
| + Dropout | 74% | 66% | 71% |
| + GlobalaveragePool | 73% | 63% | 72% |
| + Deeper model | 90% | 86% | 86% |
| + LR Scheduler | 92% | 88% | 88% |
| + AdamW | 95% | 89% | 90% |

- Adam was making large weights to fit the training data which caused overfitting and worse generalization as AdamW that applies weights decay , a penalty to the loss function to discourage large weights
- We decide to stop the training at this level

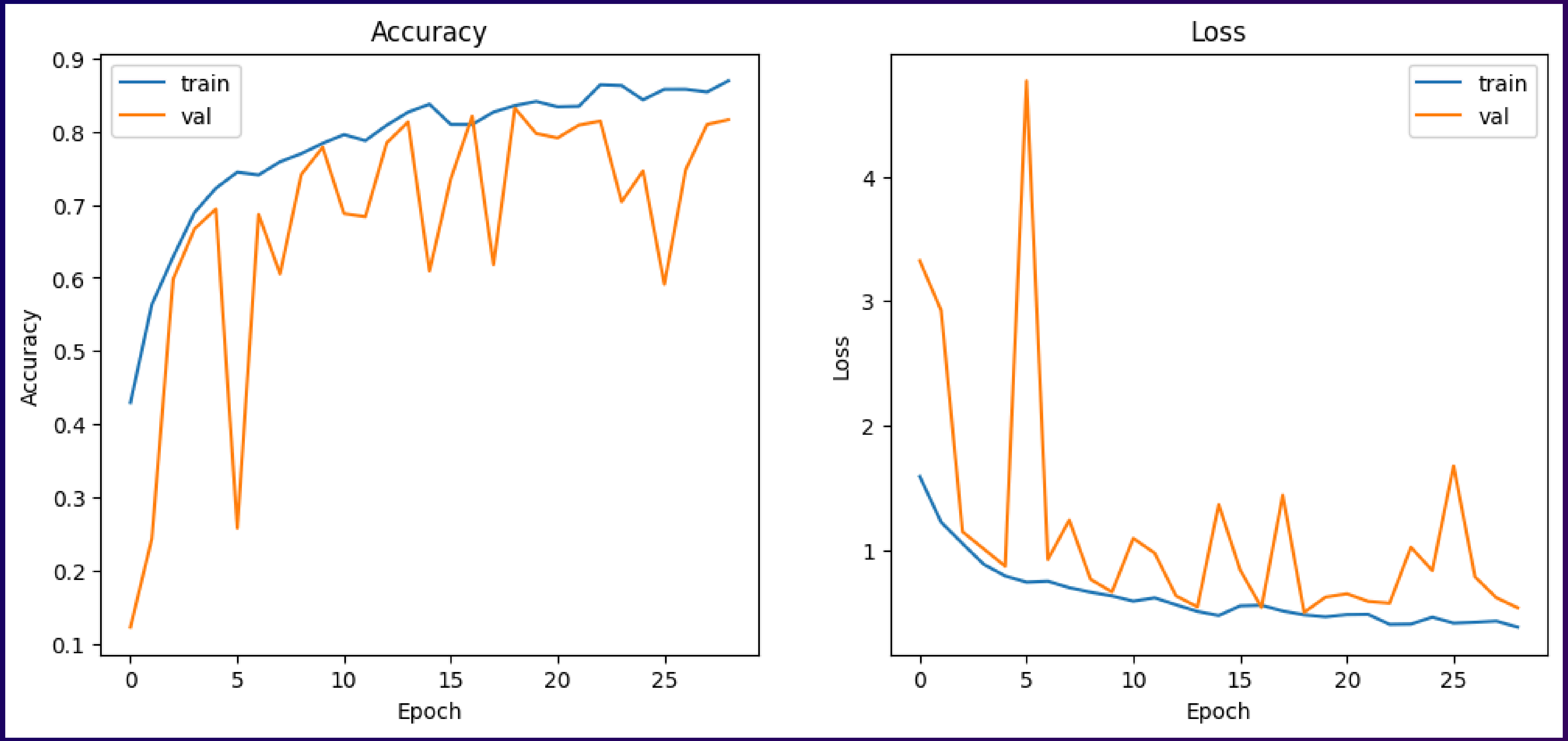
# Model with AdamW - Evaluation




- Misclassification occurs between class 3 and 5, which explains lower recall (0.75 and 0.77)
- Most classes show high precision and recall (>0.87)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.92 | 0.91 | 1000 |
| 1 | 0.94 | 0.97 | 0.95 | 1000 |
| 2 | 0.86 | 0.87 | 0.86 | 1000 |
| 3 | 0.80 | 0.75 | 0.78 | 1000 |
| 4 | 0.89 | 0.87 | 0.88 | 1000 |
| 5 | 0.89 | 0.77 | 0.83 | 1000 |
| 6 | 0.87 | 0.95 | 0.91 | 1000 |
| 7 | 0.92 | 0.94 | 0.93 | 1000 |
| 8 | 0.94 | 0.94 | 0.94 | 1000 |
| 9 | 0.90 | 0.95 | 0.93 | 1000 |

# Model with AdamW - Evaluation




- By the end of the training, the model has reached a good balance between validation and training

- There are strong fluctuations during training, This suggests the model might be sensitive to the mini-batches

# Transfer Learning

AI poses ethical concerns such as data privacy, bias, and accountability. Transparent algorithms, regulatory frameworks, and responsible AI development are essential to ensure fairness and prevent misuse.

# inception_v3

AI is transforming the workforce by automating repetitive tasks and augmenting human capabilities. While some jobs may be displaced, new roles will emerge, emphasizing the need for upskilling and lifelong learning.

# efficentNet_v2

AI is transforming the workforce by automating repetitive tasks and augmenting human capabilities. While some jobs may be displaced, new roles will emerge, emphasizing the need for upskilling and lifelong learning.

# denseNet

AI is transforming the workforce by automating repetitive tasks and augmenting human capabilities. While some jobs may be displaced, new roles will emerge, emphasizing the need for upskilling and lifelong learning.

# Conclusions

Businesses leverage AI for data analysis, customer service automation, and process optimization. Predictive analytics help companies make informed decisions, while AI-driven marketing personalizes customer experiences, boosting engagement and revenue.

# THANK YOU

\-

\-