Image classification of fashion items

Agenda

- Introduction
- Methodology
- Results
- Demo
- Conclusions

Introduction

- Fashion-MNIST is a data set of 28x28 grayscale images representing fashion items.
- Due to its complexity, it serves as a benchmark for evaluating and comparing image classification algorithms.
- Fashion item classification has practical applications in industries such as e-commerce, retail, and fashion.
- The data set contains 60,000 training images, 10,000 test images across 10 fashion categories.

Methodology

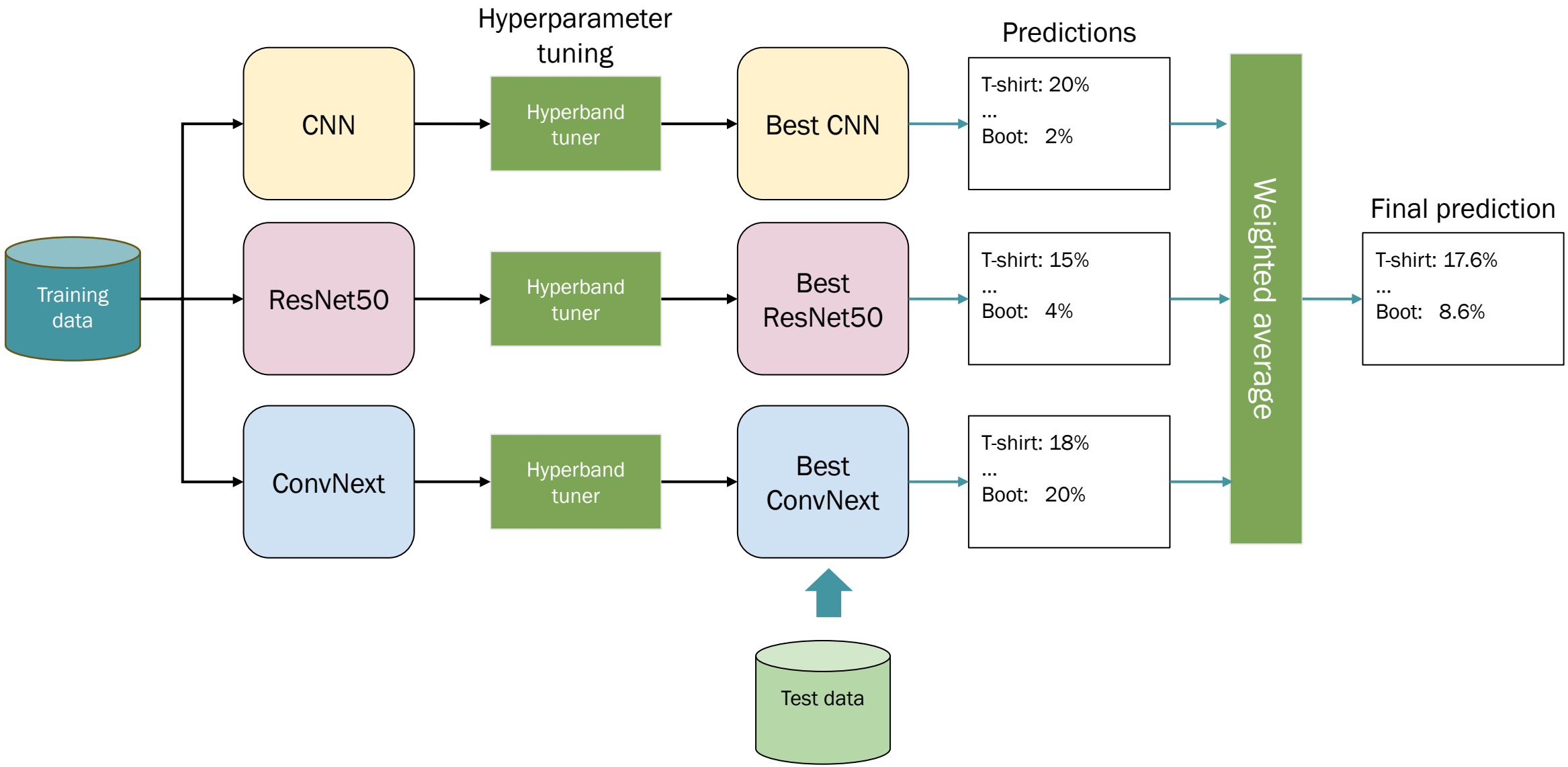
Architecture:

- Model ensemble of various neural network architectures.

Training:

- Hyperparameter tuning.
- Data augmentation.

Overview


Hyperparameter tuning
Predictions
Training data
CNN
Hyperband tuner
Best CNN
T-shirt: 20%
...
Boot: 2%
ResNet50
Hyperband tuner
Best ResNet50
T-shirt: 15%
...
Boot: 4%
ConvNext
Hyperband tuner
Best ConvNext
T-shirt: 18%
...
Boot: 20%
Weighted average
Final prediction
T-shirt: 17.6%
...
Boot: 8.6%
Test data

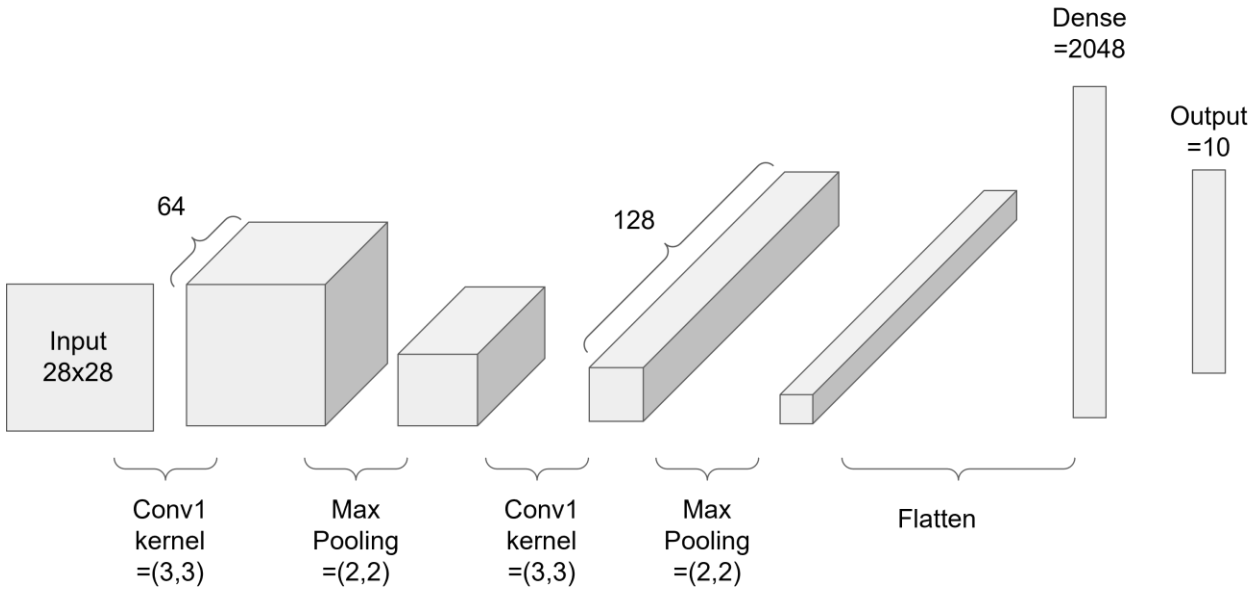
Convolutional Neural Network

CNNs are the most popular deep learning models used to extract features from images.

In our case, a custom CNN was designed. The final architecture consists of roughly 5 layers shown on the right.

The hyperparameters that need to be tuned include:

- optimizer, learning rate, beta 1 & 2, weight decay, and dropout rate.




```
All:
activation='relu'
```

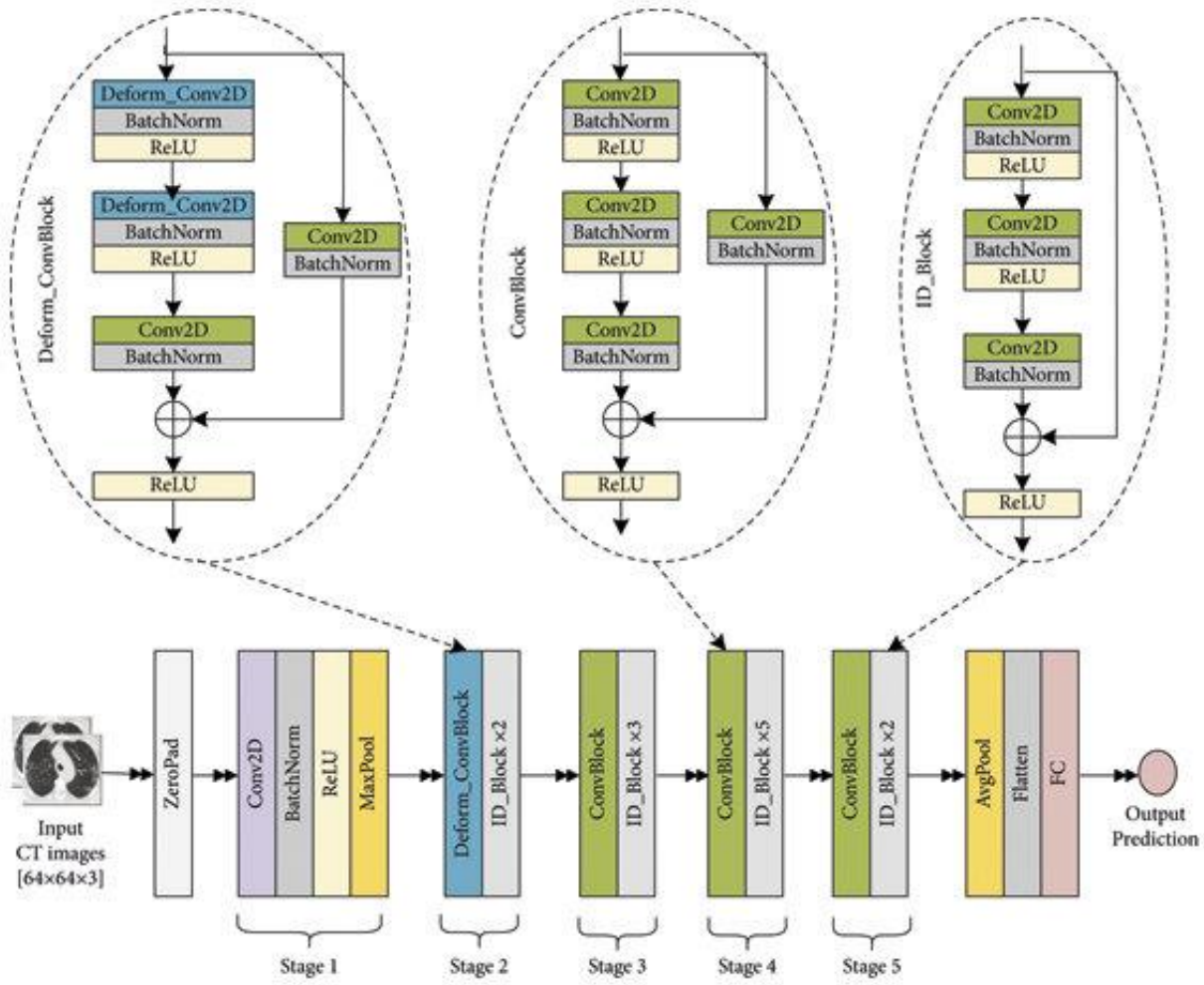
ResNet50

ResNet50 is a specific CNN architecture known for its residual connections between different stages.

In this work, we trained a ResNet50 from scratch to adapt the architecture to the problem at hand.

The hyperparameters that need to be tuned include:

- optimizer, learning rate, beta 1 & 2, weight decay, and dropout rate.



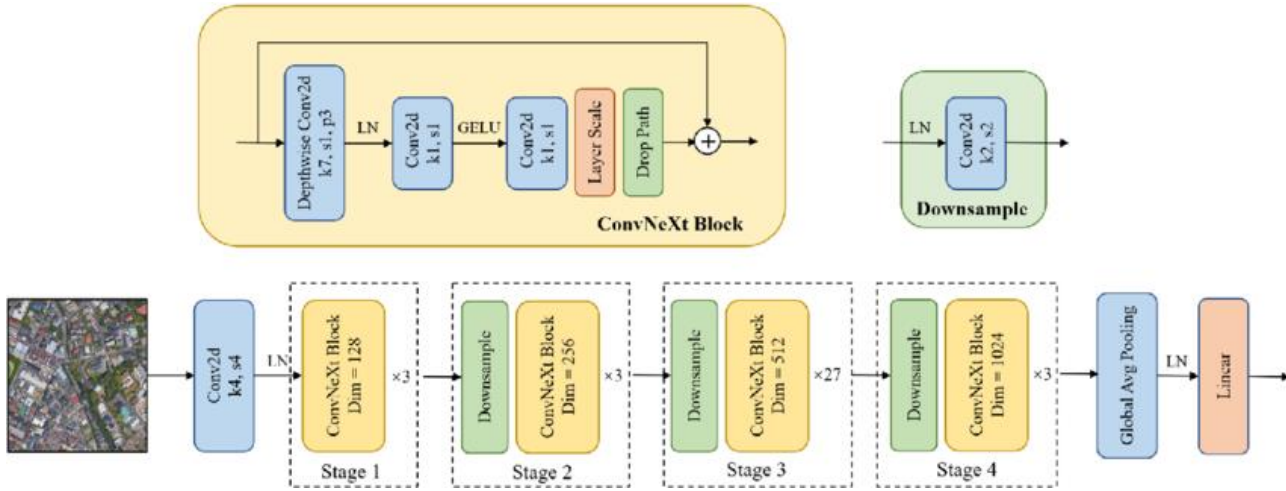
ConvNeXt

ConvNeXt is a ‘modernization’ of a standard ResNet toward the design of a vision Transformer (ViT).

Transfer learning was applied to this model, for comparison with the other models which were trained from scratch.

The hyperparameters that need to be tuned include:

- optimizer, learning rate, fine tuning layer, weight decay, and dropout rate.



Hyperparameter tuning using Hyperband

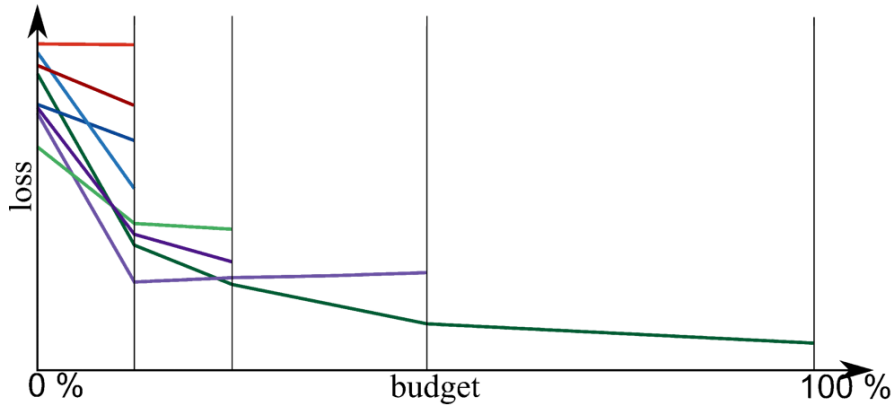
Basics:

- Hyperparameter tuning posed as a pure-exploration multi-armed bandit problem.
- Improves on the idea of successive halving.

Characteristics:

- Allocates resources (epochs) to the most promising configurations, discards the rest

Successive halving



Could we be discarding good candidates too early?

Hyperparameter tuning using Hyperband

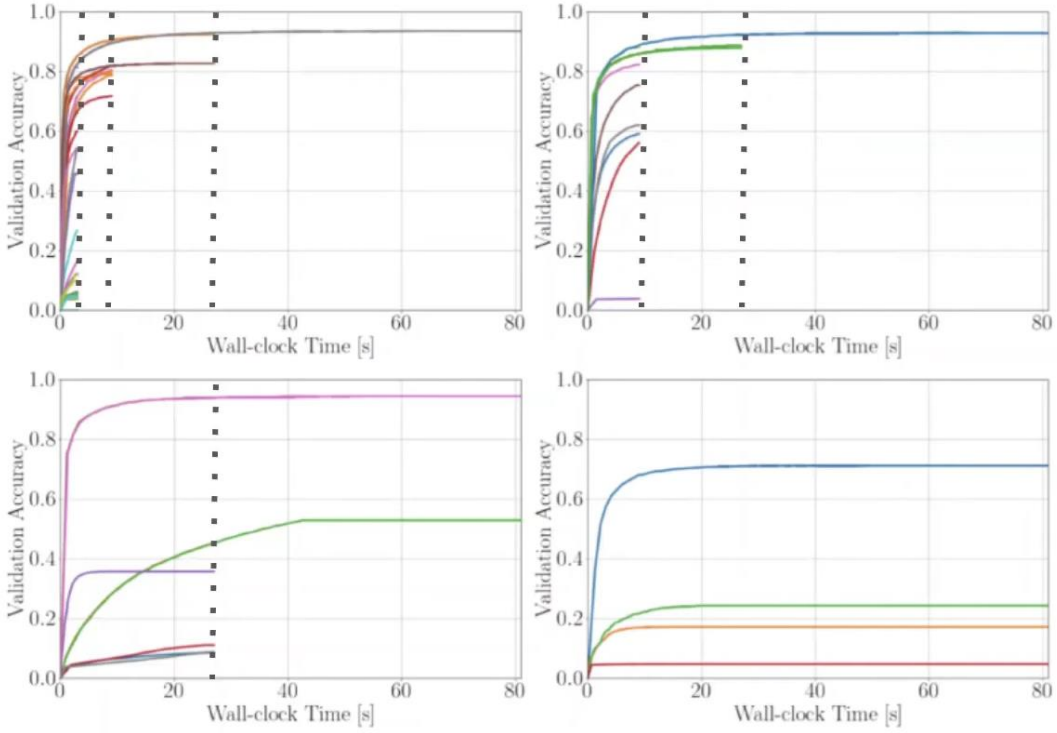
Basics:

- Hyperparameter tuning posed as a pure-exploration multi-armed bandit problem.
- Improves on the idea of successive halving.

Characteristics:

- Allocates resources (epochs) to the most promising configurations, discards the rest
- Considers that good candidates might be discarded early so it uses different brackets.

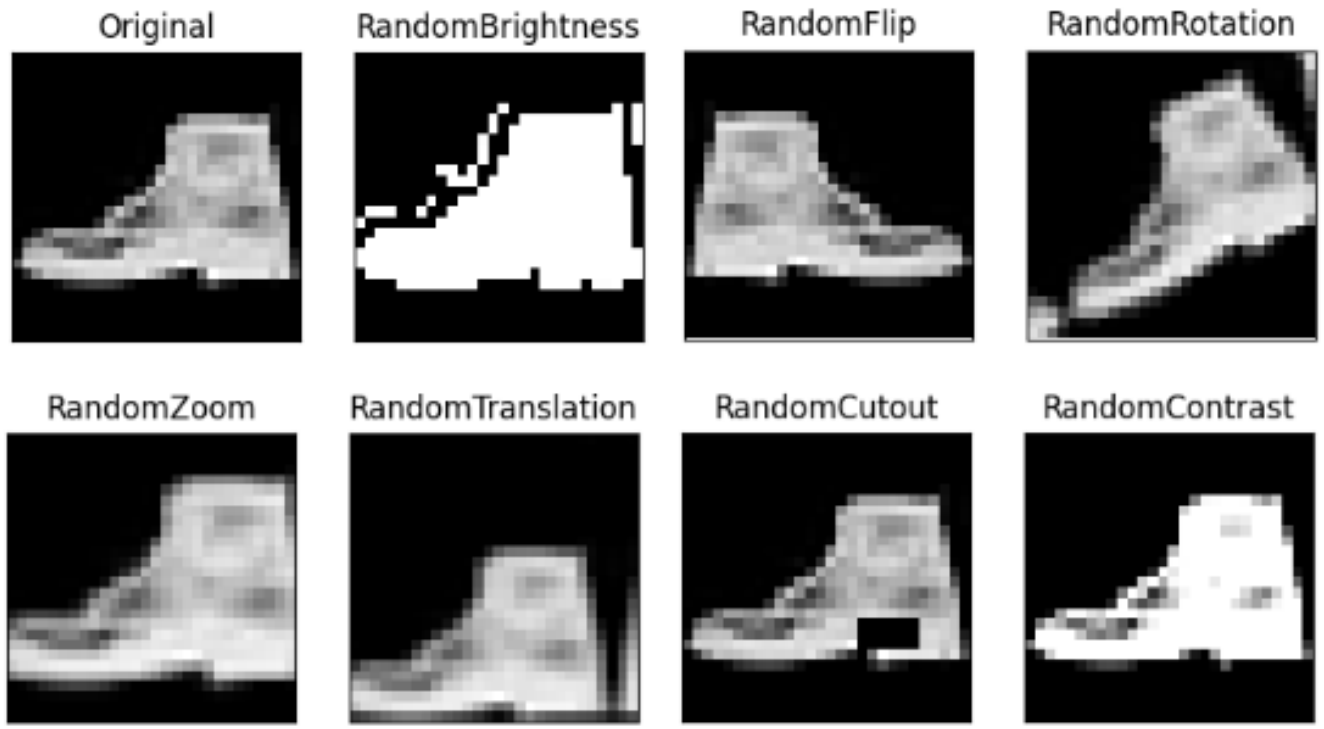
Hyperband brackets



Improving robustness through data augmentation

The following transformations to the input data were considered:

- Random Brightness
- Random Flip
- Random Rotation
- Random Zoom
- Random Translation
- Random Cutout
- Random Contrast
- Synthetic image generation (GAN)*

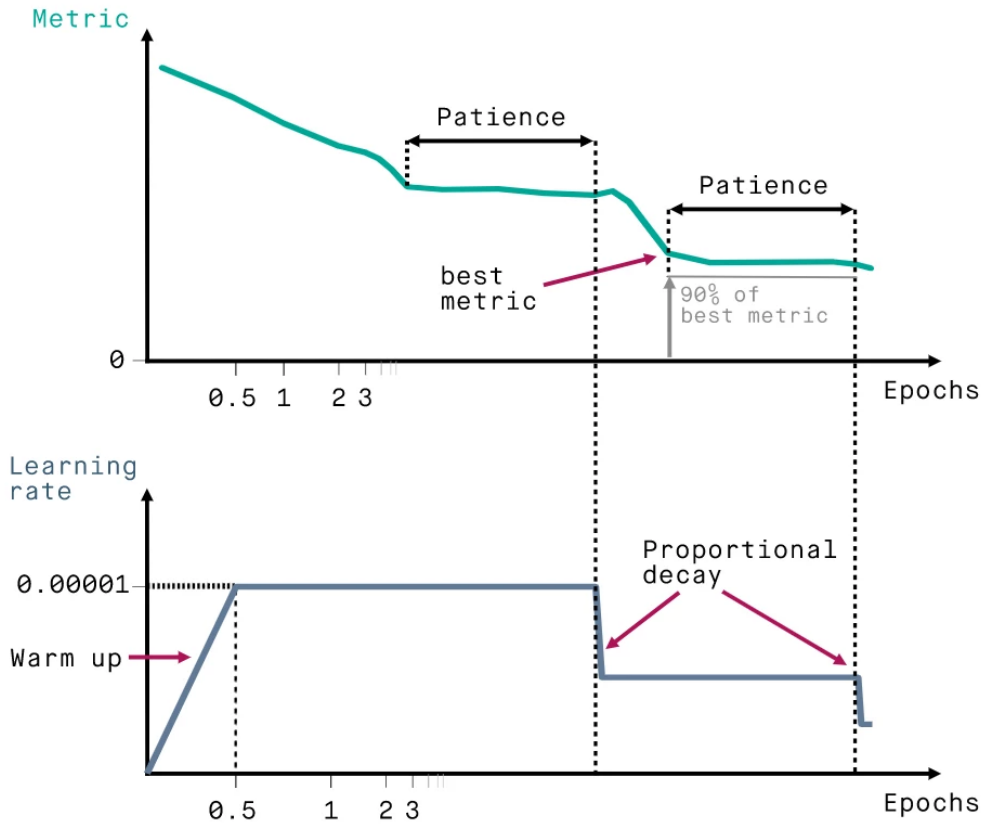


Improving training with learning rate reduction on Plateu

Adapting learning rate through training can help improve the model's performance and avoid local minima.

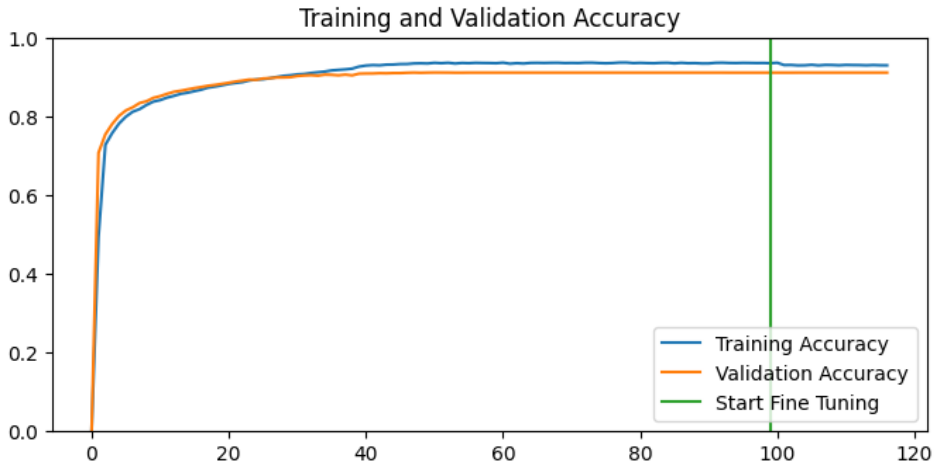
A common approach is to reduce the learning rate once the performance stops improving by a few epochs (called patience).

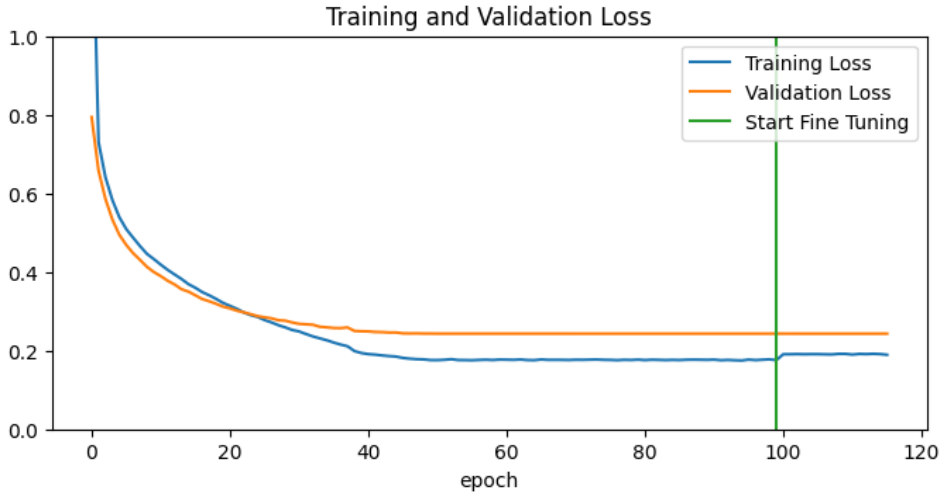
We use this technique with a patience of 1 epoch.



Results

Simple CNN performance





Learning curves

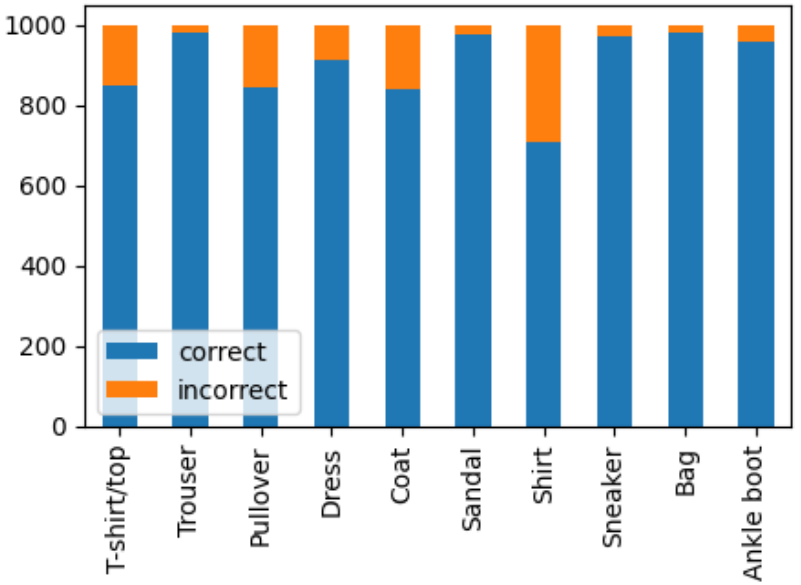
Summary

Training accuracy	Validation accuracy	Testing accuracy
0.9358	0.9107	0.9016

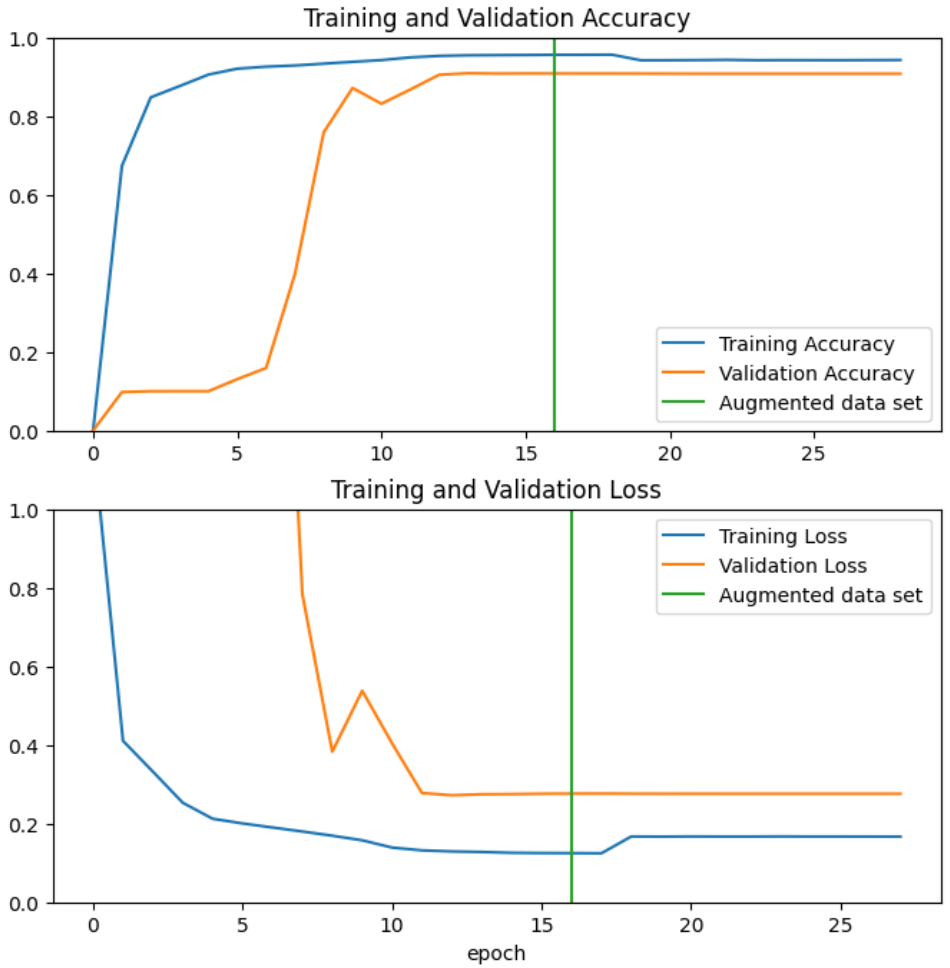
Best parameters

```
'optimizer': 'nadam',
'learning_rate': 0.001,
'beta_1': 0.9,
'beta_2': 0.99,
'weight_decay': 0.0,
'dropout': 0.1
```

Performance on testing set



ResNet50 performance



Learning curves

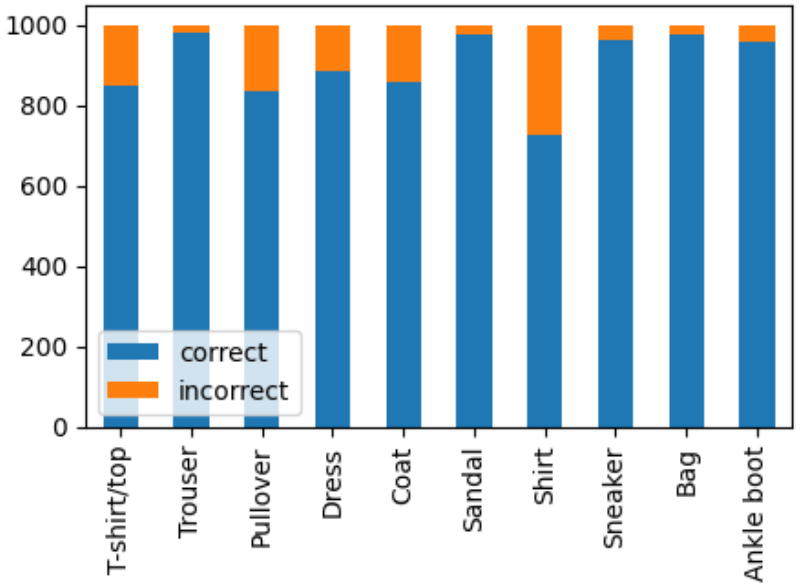
Summary

Training accuracy	Validation accuracy	Testing accuracy
0.9622	0.9104	0.9006

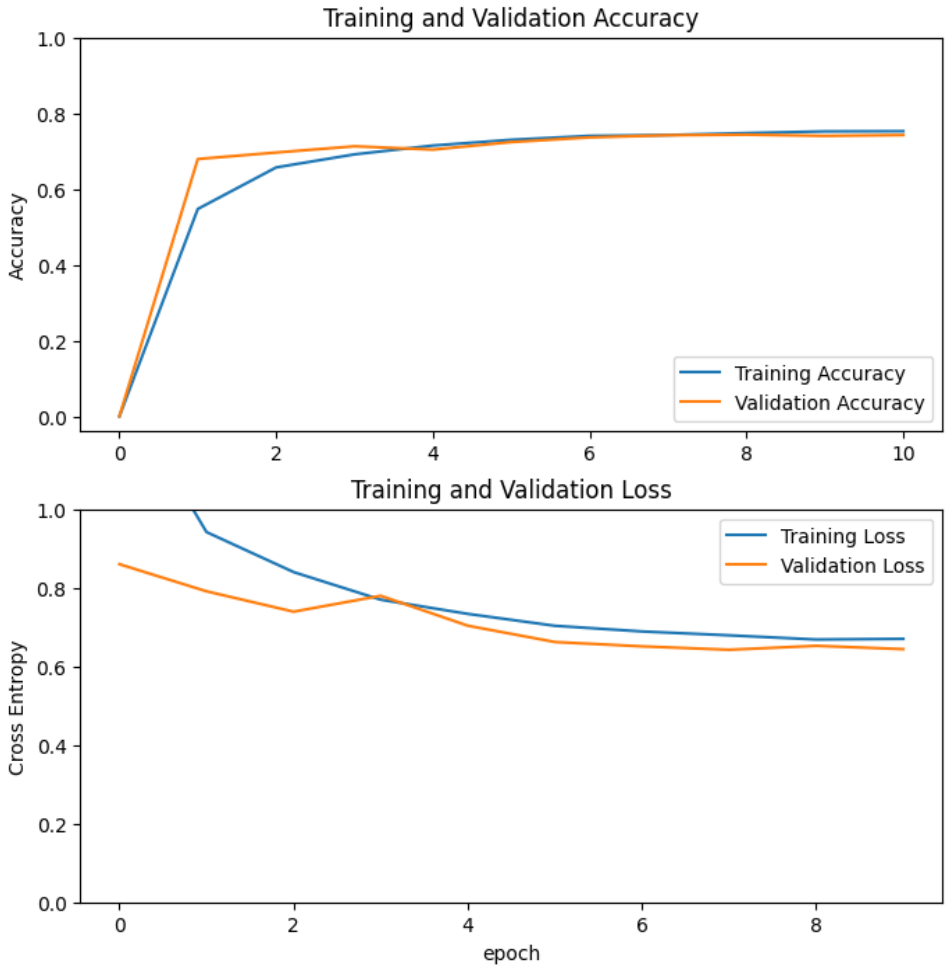
Best parameters

```
'optimizer': 'nadam',
'learning_rate': 0.01,
'beta_1': 0.9,
'beta_2': 0.9999,
'weight_decay': 1e-05,
'dropout': 0.1,
```

Performance on testing set



ConvNeXt performance



Learning curves

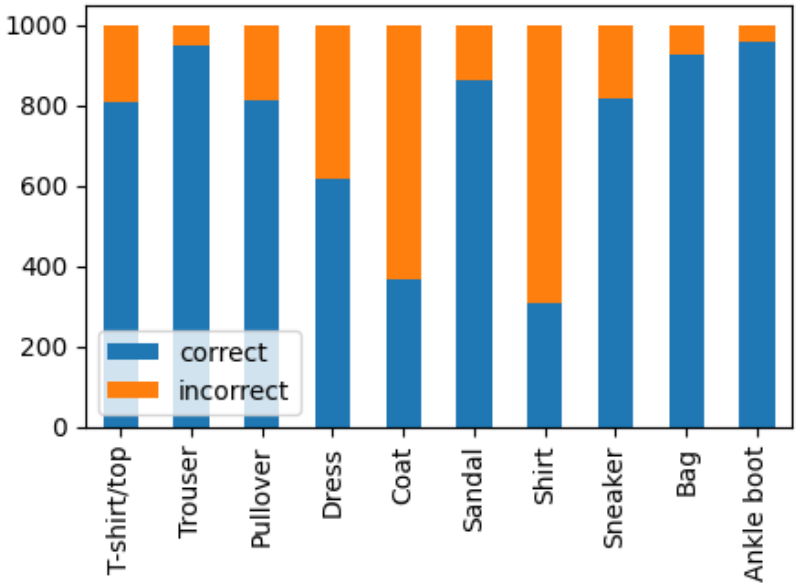
Summary

Training accuracy	Validation accuracy	Testing accuracy
0.7459	0.7425	0.7423

Best parameters

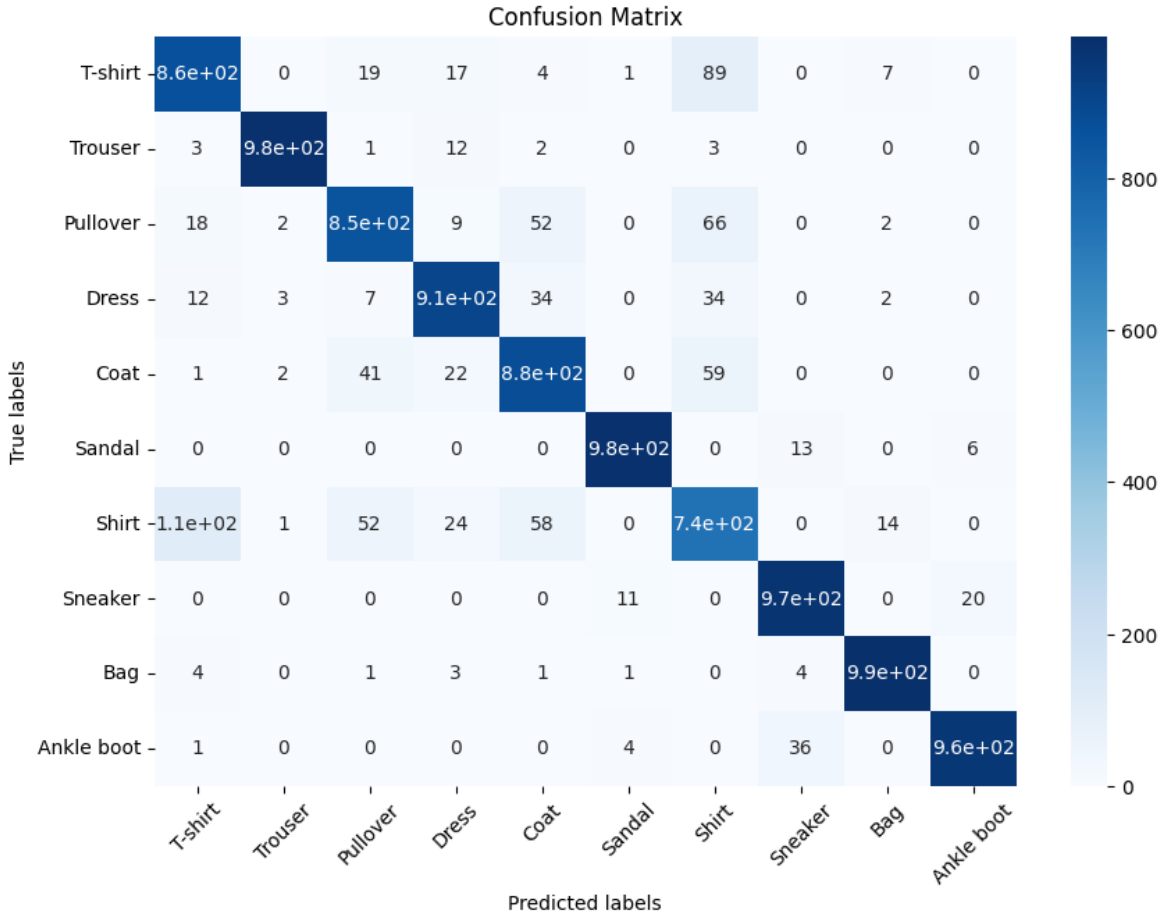
```
'optimizer': 'nadam',
'learning_rate': 0.001,
'fine_tune_at': 143,
'weight_decay': 1e-04,
'lr_schedule':
94_step_0_9
'dropout': 0.1,
```

Performance on testing set



Ensemble performance on testing set

Model	Testing set	Weight
CNN	0.9016	0.45
ResNet50	0.9007	0.45
ConvNeXt	0.7423	0.10
Ensemble	0.911	

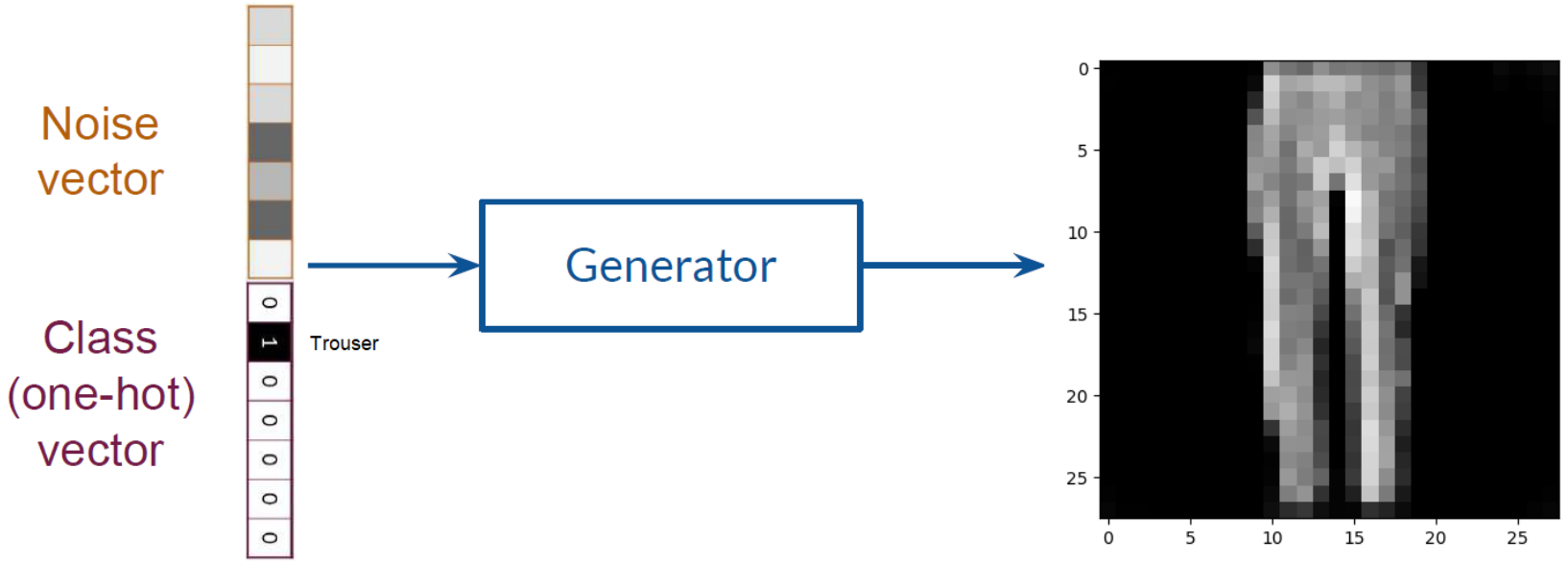


Our model confuses T-shirts with Shirts the most

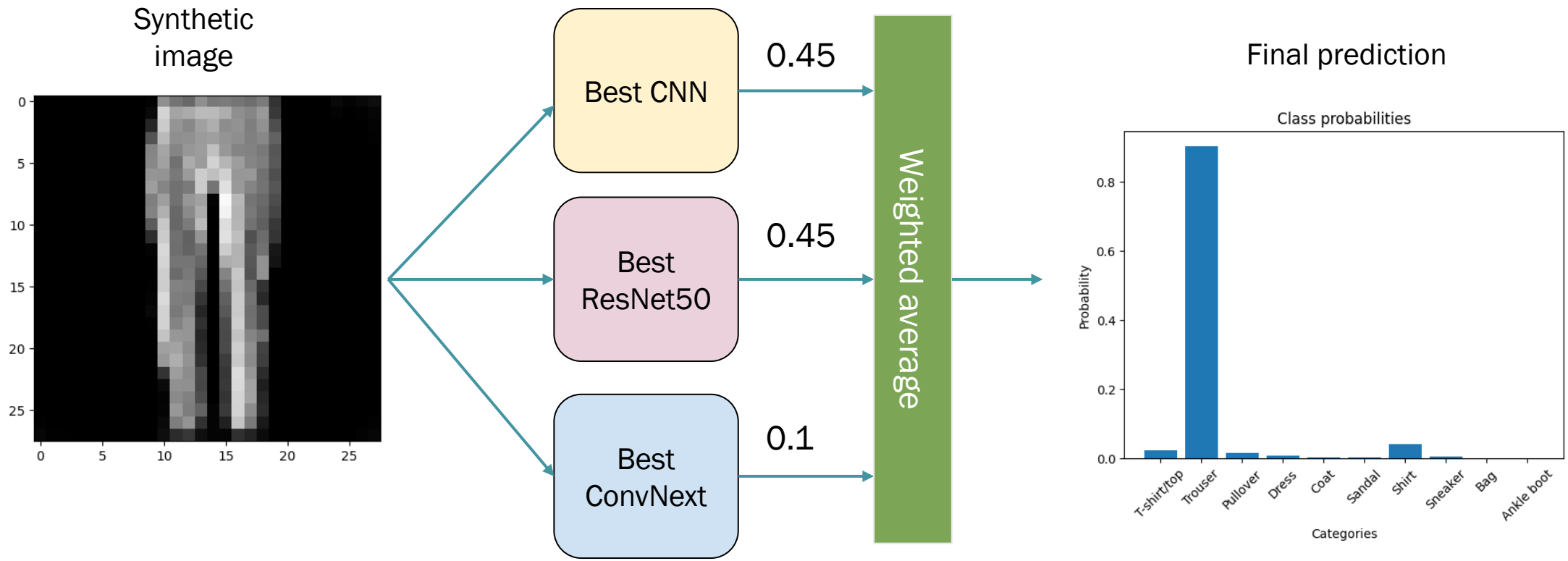
DEMO

Performance against unseen fashion item samples

Synthetic image generation using GAN



Demo


Synthetic image
Best CNN
0.45
Best ResNet50
0.45
Best ConvNext
0.1
Weighted average
Final prediction
Class probabilities
Probability
Categories
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot

Conclusions

- Hyperparmeter tuning through Hyperband is useful for balancing resources and training time.
- Training models from scratch appears to be better than using transfer learning likely due to the differences in the images used for pretraining models.
- Data augmentation through GAN needs to be further tuned to create ‘difficult’ samples otherwise the benefits are small.
- The model confuses t-shirts with shirts the most, for some samples it is even hard for a human to distinguish.
- Model ensemble provides slightly better performance but for this problem a traditional CNN could be a simpler and viable approach.

Questions?