Cloud Computing for High Dimensional Data : Final Project Presentation

Image Classification of Clothing

SIVAKUMAR Fabrice, OULIE Yacine

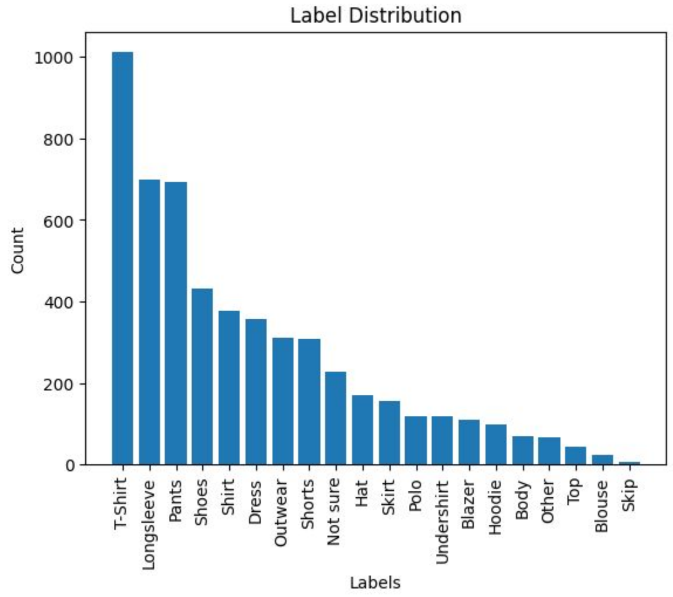
Outline

Dataset Presentation

Dataset Presentation

- A dataset of over 5,000 images of clothing items under a public domain license from Kaggle
- Covers 20 different types of clothes and can be used for various purposes, including commercial use
- Data collection involved three sources: Toloka (crowdsourcing platform), networking through social media, and Tagias (a company specializing in data collection)
- Toloka contributed 1,400 images (28% of the dataset), but required extra analysis to filter out irrelevant or duplicated images.
- Networking through social media resulted in 32 contributors submitting 600 images.
- Tagias contributed the majority, providing 3,000 images (60% of the dataset), and ensured the authenticity of the images through an internal validation process.
- Manual labeling of the images was done using an IPython widgets-based annotation tool.
- Labeling mistakes were corrected using a simple neural network approach.

Dataset Presentation



Imbalanced dataset => Need to perform oversampling/undersampling

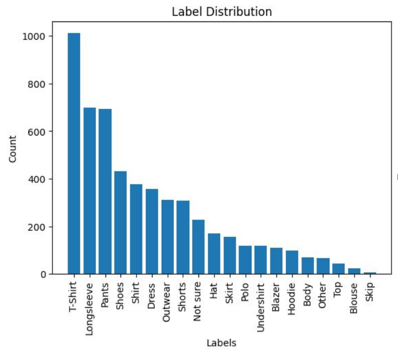
Code Structure

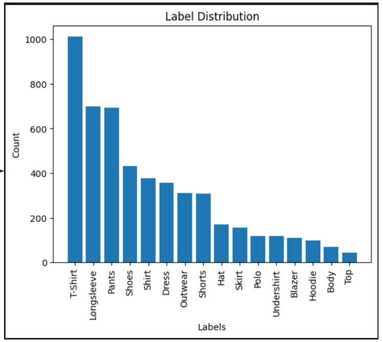
Code Structure

- Data Preprocessing
- Training part involving a training set and validation set (fine tune the learning rate)
- Test part gives a percentage of the model accuracy
- Predicted vs Actual Labels part

Data Preprocessing

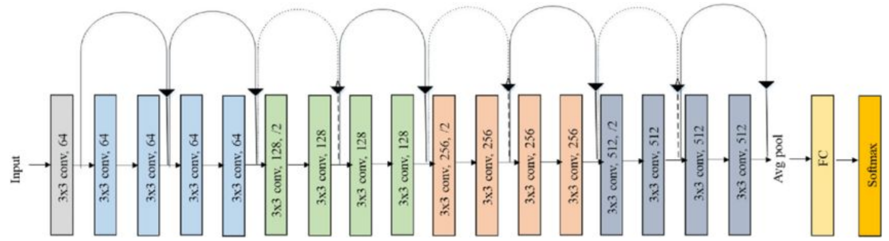
- Elimination of corrupted images from the dataset.
- Elimination of 2 classes('Others' and 'Not Sure') for image classification.
- Exclusion of 2 classes('Skip' and 'Blouse') due to unclear and difficult-to-recognize images.
- Creation of a Python dictionary for label encoding, assigning a unique number to each label.



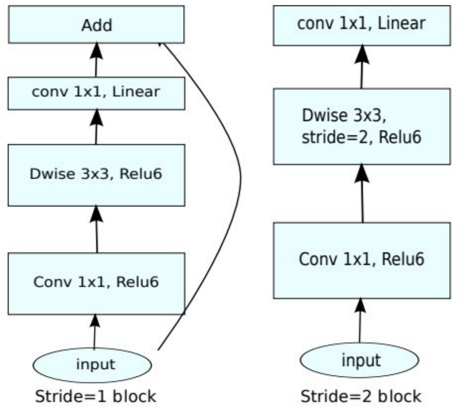


Training

Resnet18



MobileNetV2



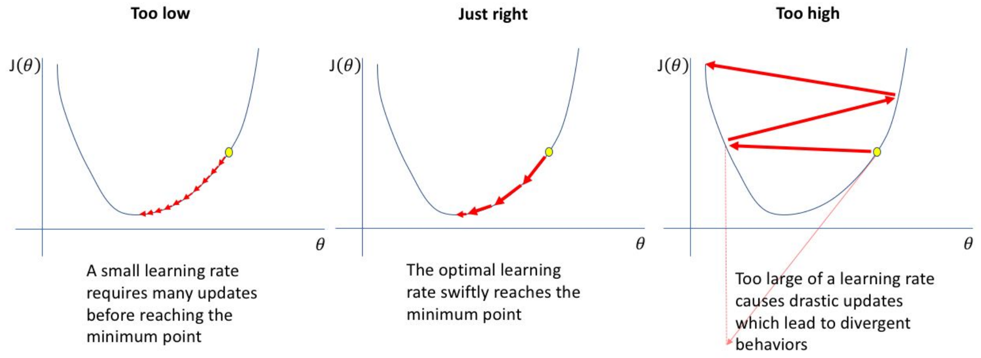
(d) Mobilenet V2

Predicted vs Actual Labels

- We take 10 random images from the test set and compare the predicted label and the actual labels
- Green when it's the same
- Red otherwise

Importance of choosing the right learning rate

Importance of choosing the right learning rate

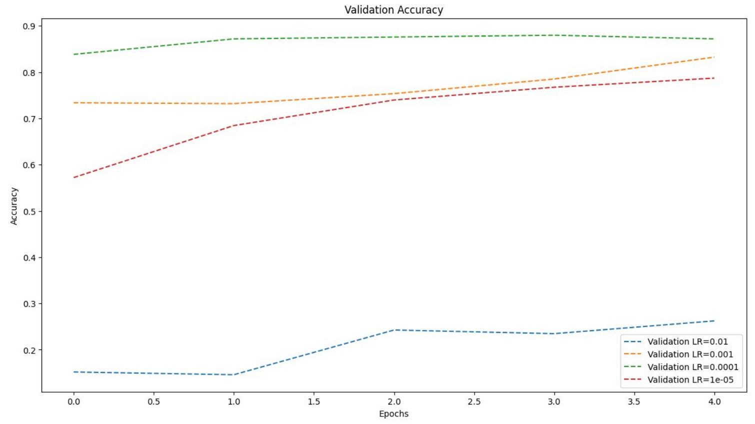


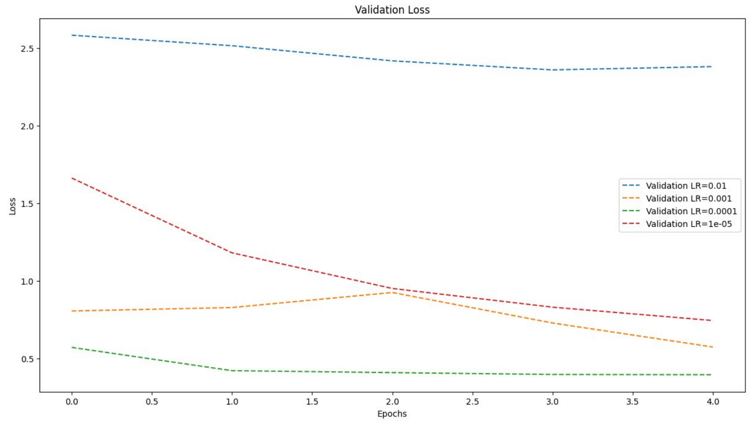
A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

Too large of a learning rate causes drastic updates which lead to divergent behaviors

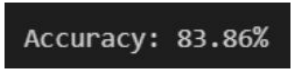
Importance of choosing the right learning rate

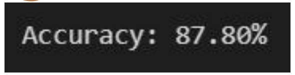


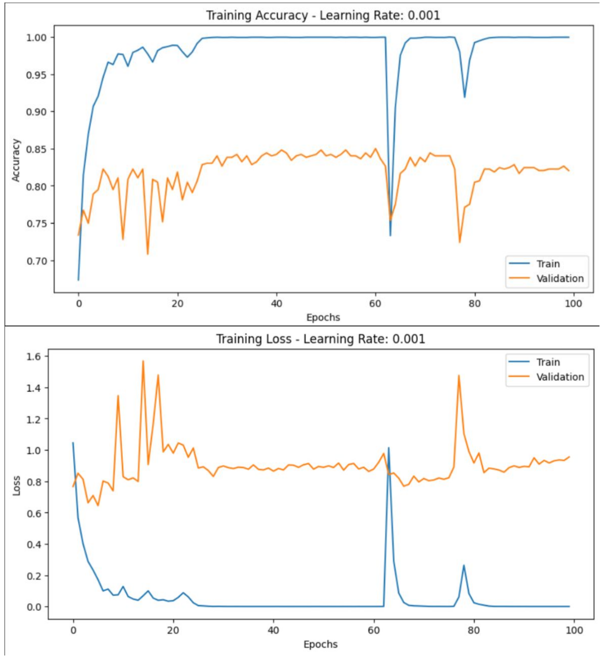


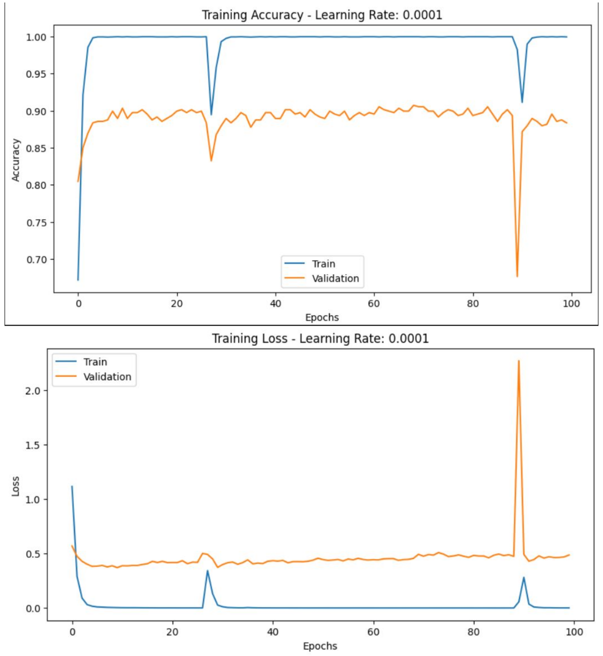
The best learning rate among these four rates to choose is 1e-4

Importance of choosing the right learning rate


```
Accuracy: 83.86%
```


```
Accuracy: 87.80%
```

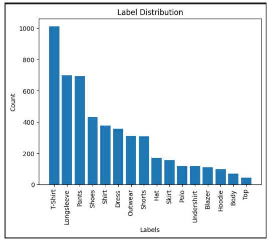




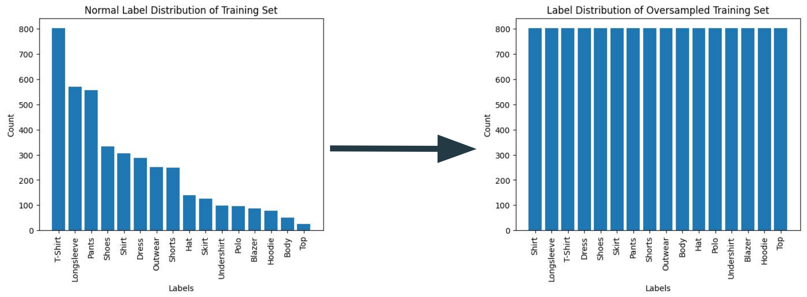
Oversampling/Undersampling

Oversampling/Undersampling

- To tackle the imbalance issue in the dataset that can affect the training of the model :
  - Oversampling
  - Undersampling
  - Oversampling + Undersampling

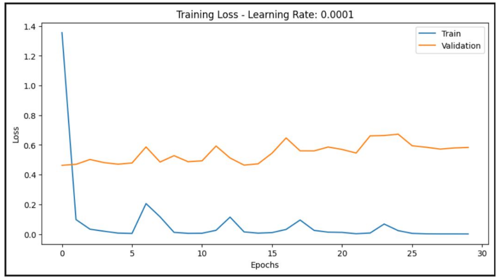


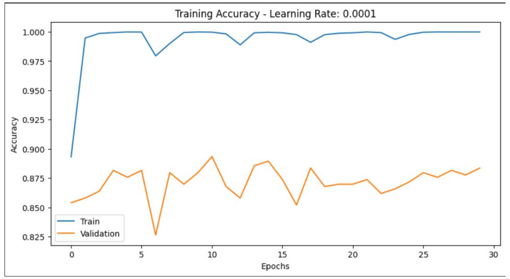
- Oversampling
  - Oversample the minority classes in the training set
  - ‘RandomOverSampler’ function from imblearn library
  - Use of ‘not majority’ sampling strategy => all except the majority class
  - Select instances randomly from the minority classes and create duplicates of the selected instances.

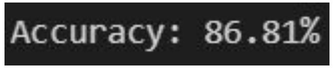


Oversampling / Undersampling

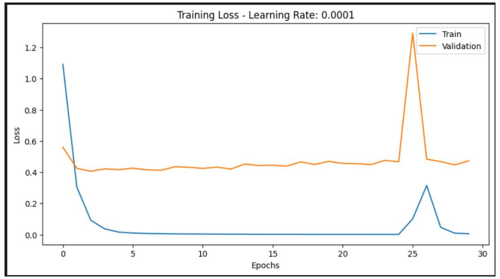
With Oversampling

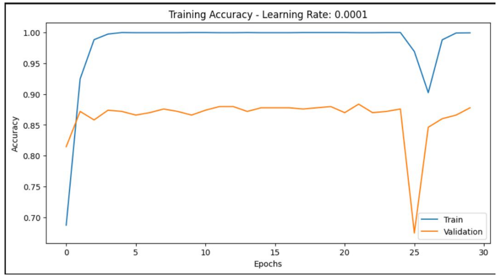


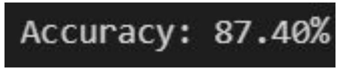



```
Accuracy: 86.81%
```

Without Oversampling



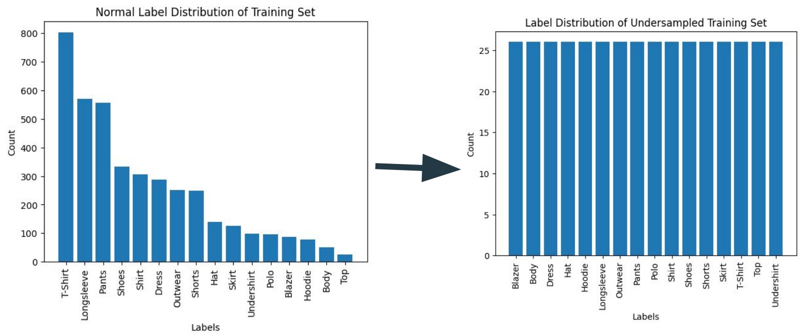



```
Accuracy: 87.40%
```

- Advantages :
  - No more big fluctuations
  - Less losses
- Drawbacks:
  - Accuracy in test set is smaller with oversampling but with 100 epochs it could be different
  - Longer training time

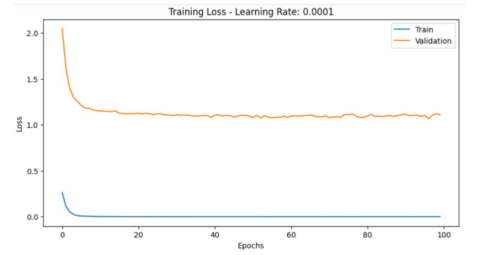
Oversampling/Undersampling

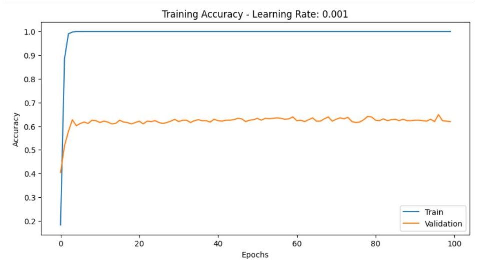
- Undersampling
  - Undersample the majority classes in the training set
  - ‘RandomUnderSampler’ function from imblearn library
  - Use of ‘not minority’ sampling strategy => all except the minority class
  - Select instances randomly from the majority classes to create a subset, discarding the rest.



Oversampling/Undersampling

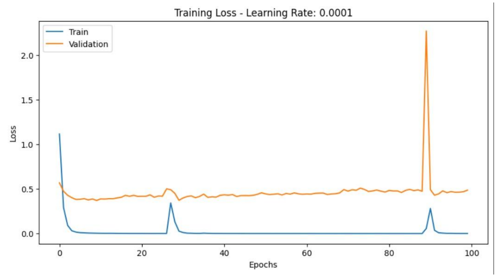
With Undersampling

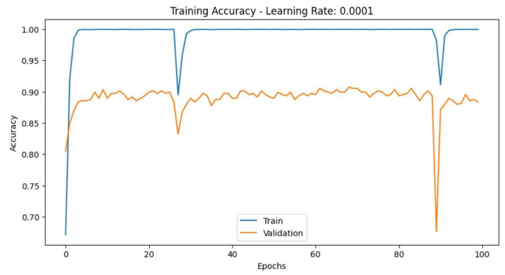




```
Accuracy: 66.73%
```

Without Undersampling



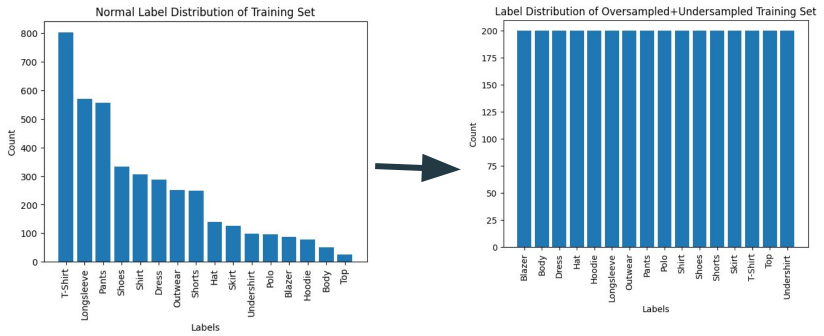


```
Accuracy: 87.80%
```

- Advantages :
  - No more big fluctuations
  - Take less time to train
- Drawbacks:
  - Accuracy in test set is smaller with undersampling
  - Big losses and small accuracy in validation set => lost important information of the majority classes

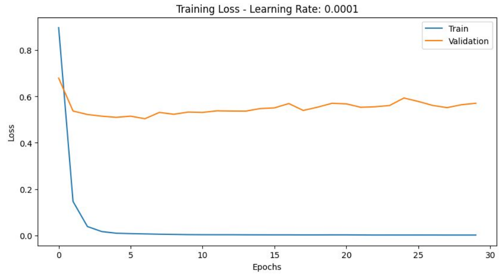
Oversampling/Undersampling

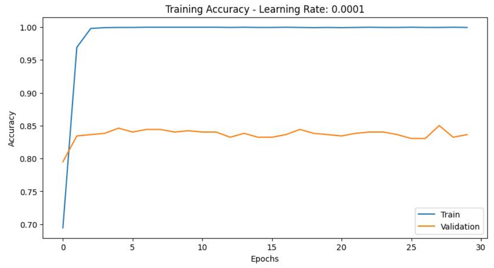
- Oversampling + Undersampling
  - Oversample the minority classes in the training set
  - Undersample all classes in the training set to have a smaller amount in each class

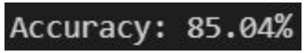


Oversampling/Undersampling

With Oversampling + Undersampling

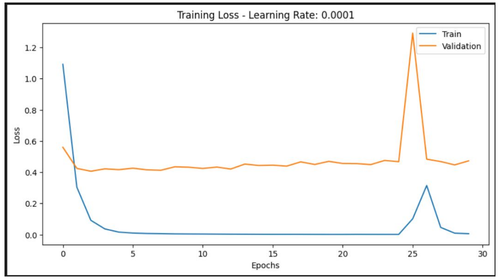


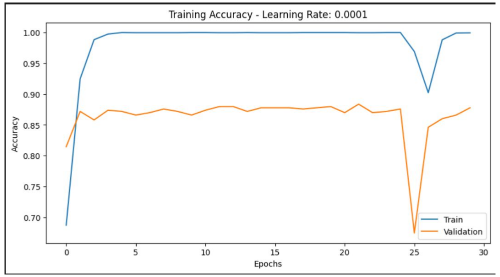


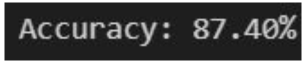


```
Accuracy: 85.04%
```

Without Oversampling +Undersampling







```
Accuracy: 87.40%
```

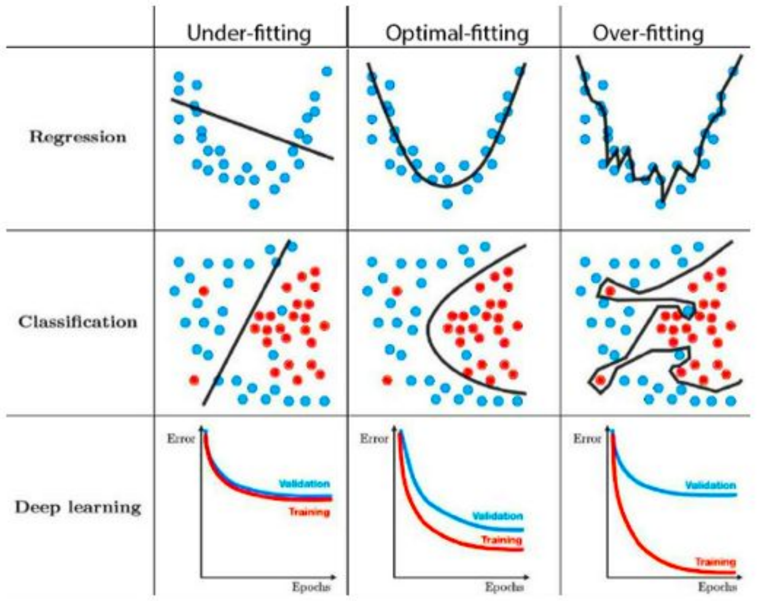
- Advantages :
  - No more big fluctuations
  - Less losses depending on the number of elements we choose for each class
- Drawbacks (depending on the number of elements we choose for each class):
  - Accuracy in test set is smaller with undersampling
  - Big losses and small accuracy in validation set => lost important features of the majority classes
  - Can take more time to train the model depending on the number of elements we choose for each class
- Oversampling is the best solution to generalize a model

Importance of preventing overfitting

Importance of preventing overfitting

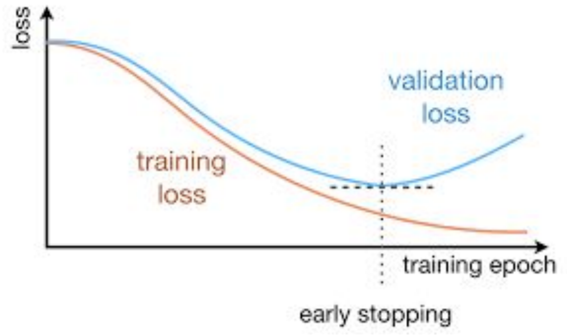
Overfitting:

- Definition:
  - Model learns training data too well, capturing noise and outliers.
- Characteristics:
  - Low training error but high validation error.
  - Model overly complex.
- Risks:
  - Poor generalization to new/unseen data.
  - Sensitivity to noise in training data.
  - Limited model applicability.



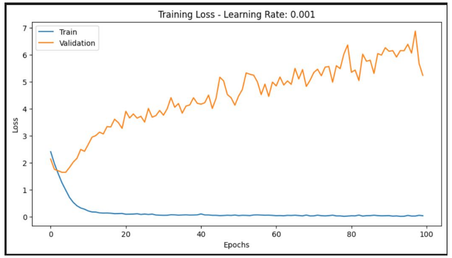
Early Stopping

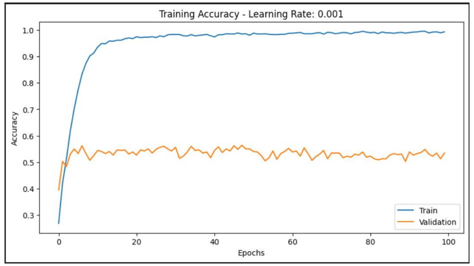
- Purpose:
  - Prevent overfitting during model training.
- Mechanism:
  - Monitors a specified metric (e.g., validation loss) during training.
  - Stops training once the metric stops improving or starts degrading.
  - Helps find the optimal point to halt training, avoiding overfitting.
- Implementation
  - Compare last validation loss with the best validation loss (`best_val_loss`).
  - Update or Increment:
    - If current validation loss is better, update `best_val_loss` and reset counter (`epochs_without_improvement`).
    - If not, increment the counter.
  - If counter surpasses patience threshold (`patience`), exit training loop.



Early Stopping

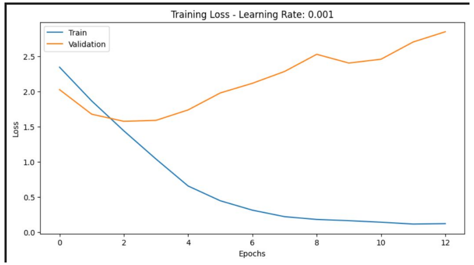
An example from our experiments where the Early Stopping was useful (our model training) :

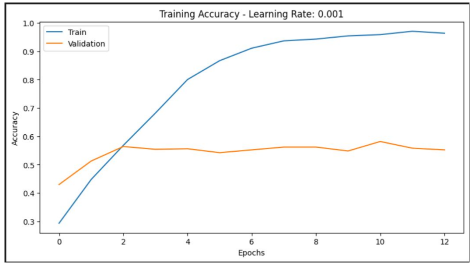




```
Accuracy: 47.05%
```

Without Early Stopping

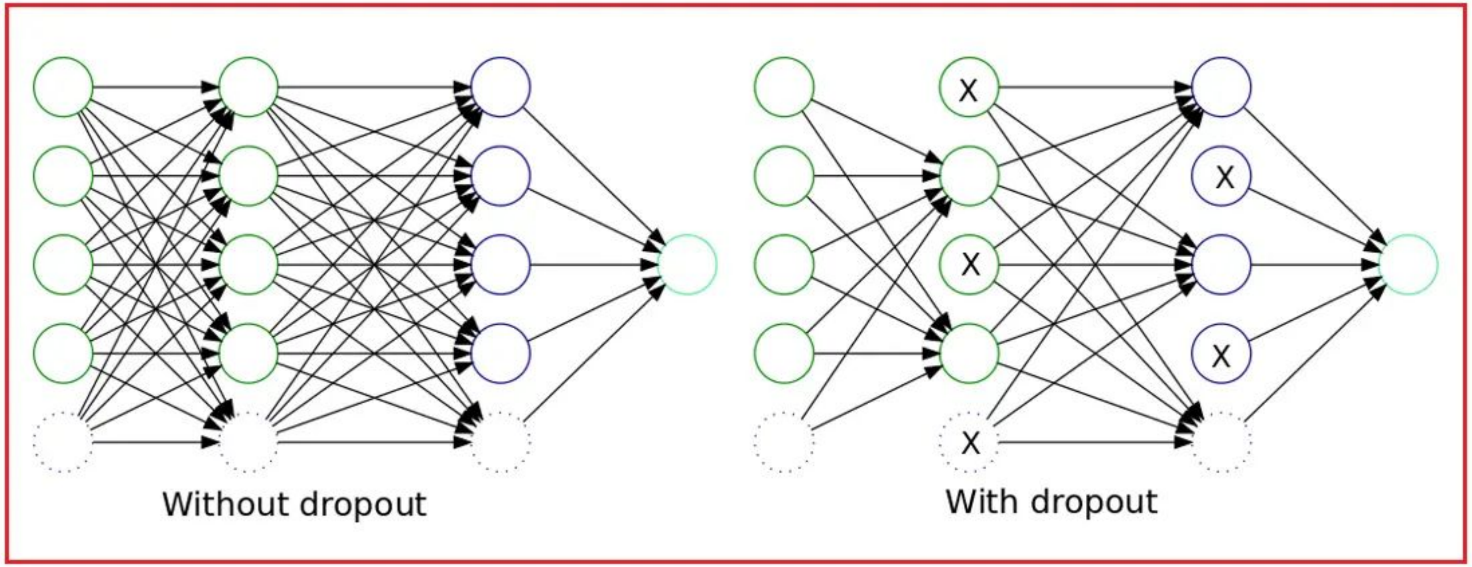




```
Accuracy: 52.95%
```

With Early Stopping

Dropout layer



Conclusion & Future Work

Conclusion & Future Work

- Successful Results with Pretrained Models: Achieved excellent results, particularly with the ResNet18 pretrained model and to a lesser extent with MobileNetV2. The primary focus of our testing and experimentation was on ResNet18.
- Effective Handling of Imbalanced Dataset: Identified oversampling as the most effective method to address fluctuations resulting from the imbalanced nature of the dataset. Oversampling proved beneficial in improving the model's ability to generalize across different classes.
- Challenges with Custom Model: Attempted to create a custom model, but encountered performance issues, reflected in a poor accuracy. However, the implementation of early stopping proved crucial in preventing overfitting and stabilizing the training process.
- Recommendations for Future Work: Suggested future work involves exploring more complex architectures and incorporating dropout layers. With additional time and resources, it is expected that these enhancements could lead to improved model performance

Thank you for your attention !