## Model:

1. ResNet50 as the base model
2. + adding custom layers:
    - GlobalAveragePooling2D:
    - Dense(1024, activation='relu')
    - Dense(num_classes, activation='softmax')
3. Adam optimizer with a learning rate of 0.001 and categorical cross-entropy as the loss function

**Pre Preprocessing**:

- rescaled to the range [0, 1] by dividing pixel values by 255.
- Data augmentation is applied through random transformations:
    - Shear range: 20% random shearing of images.
    - Zoom range: 20% random zooming.
    - Horizontal flipping: Randomly flipping images horizontally.

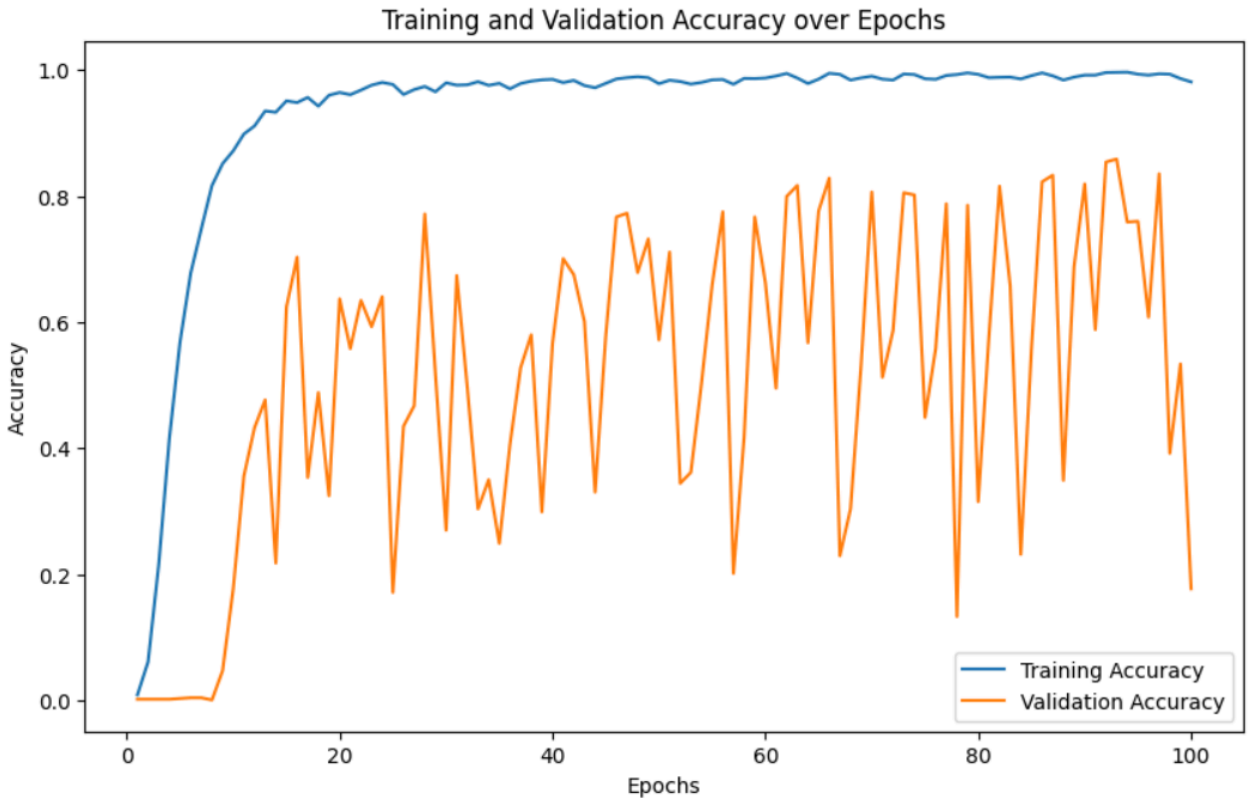
**Dataset Split**

- Training Set: 80% of the original dataset.
- Validation Set: 10% of the original dataset.
- Test Set: 10% of the original dataset.

100 epoches.

**Graph:**

```
Epoch 100/100
108/108 ━━━━━━━━━━━━━━━━━━━━ 84s 733ms/step - accuracy: 0.9837 - loss: 0.0764 - val_accuracy: 0.1775 - val_loss: 15.9916
```




**Observations:**

1. Training Accuracy:
   - The training accuracy increased consistently and reached near-perfect accuracy (close to 100%) within the first 20 epochs.
   - After that, the accuracy plateaus and remains high and stable throughout the remaining epochs. So we can think the **model is fitting well to the training data.**
2. Validation Accuracy (Orange Line):
   - The validation accuracy **fluctuated drastically** throughout the training process. It goes up and down significantly, which indicates instability in the model's performance on unseen (validation) data.
   - At certain points, the validation accuracy reaches high values, but these peaks are not sustained, and the accuracy dropped quickly in subsequent epochs.
   - The validation accuracy does not stabilized, may be due to overfitting or may be some issues with the model's generalization.

- Overfitting & Validation Instability  may be due to **Insufficient data**: More diverse data augmentation might help the model see variations and generalize better.

One of the main reasons could be the high learning rate and the complexity of the ResNet model

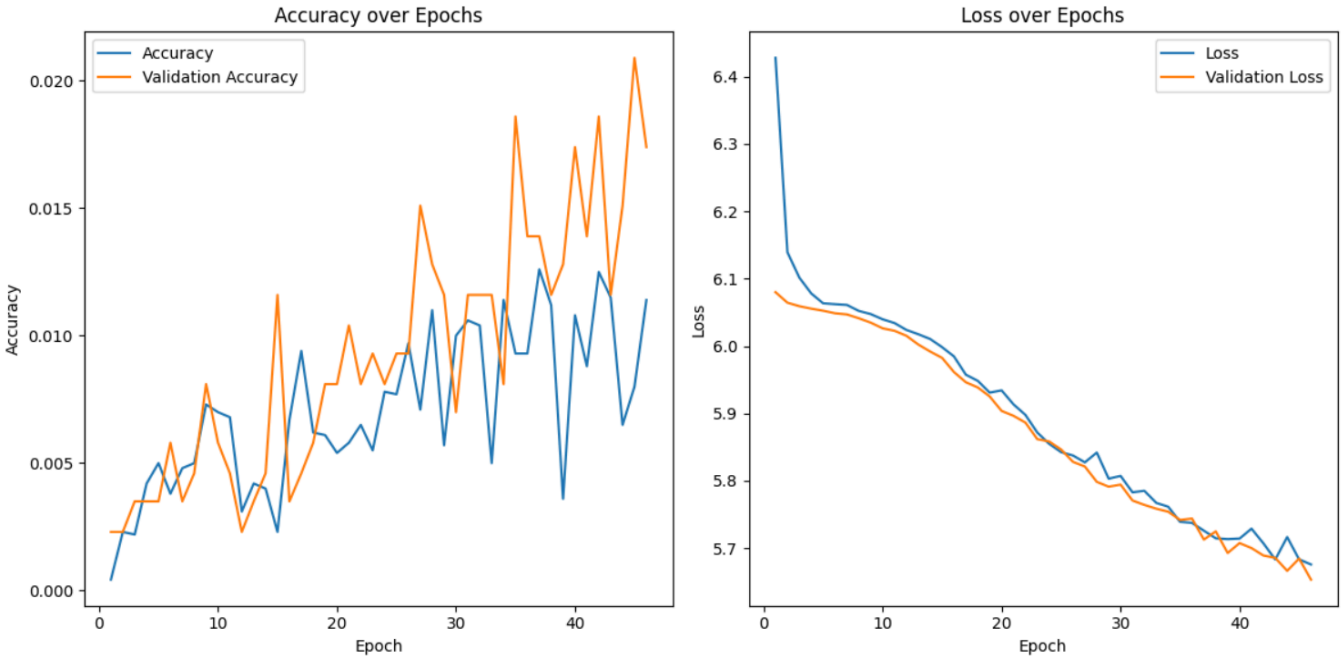
**Important changes made:**

- Added New Augmentation Techniques:
  - rotation_range=30: Random rotations of the image up to 30 degrees.
  - width_shift_range=0.2: Random horizontal shifts by 20% of the width.
  - height_shift_range=0.2: Random vertical shifts by 20% of the height.
  - brightness_range=[0.8, 1.2]: Random brightness changes, adjusting the brightness within the range of 80% to 120%.
- Added Dropout Layer (Dropout(0.5)).

The learning rate has been reduced from 0.001 to 0.0001

Added EarlyStopping Callback.

Added ReduceLROnPlateau Callback**:** This reduces the learning rate by a factor of 0.2 if the validation loss does not improve for 5 epochs, with a minimum learning rate set to 1e-6.




- **Training Accuracy & Validation Accuracy** shows a slow and steady increase over time, but it's very low.
- Despite the fluctuations, validation accuracy does not significantly diverge from training accuracy, suggesting there's no major overfitting.
- The loss decreases steadily, indicating that the model is learning and minimizing the error.

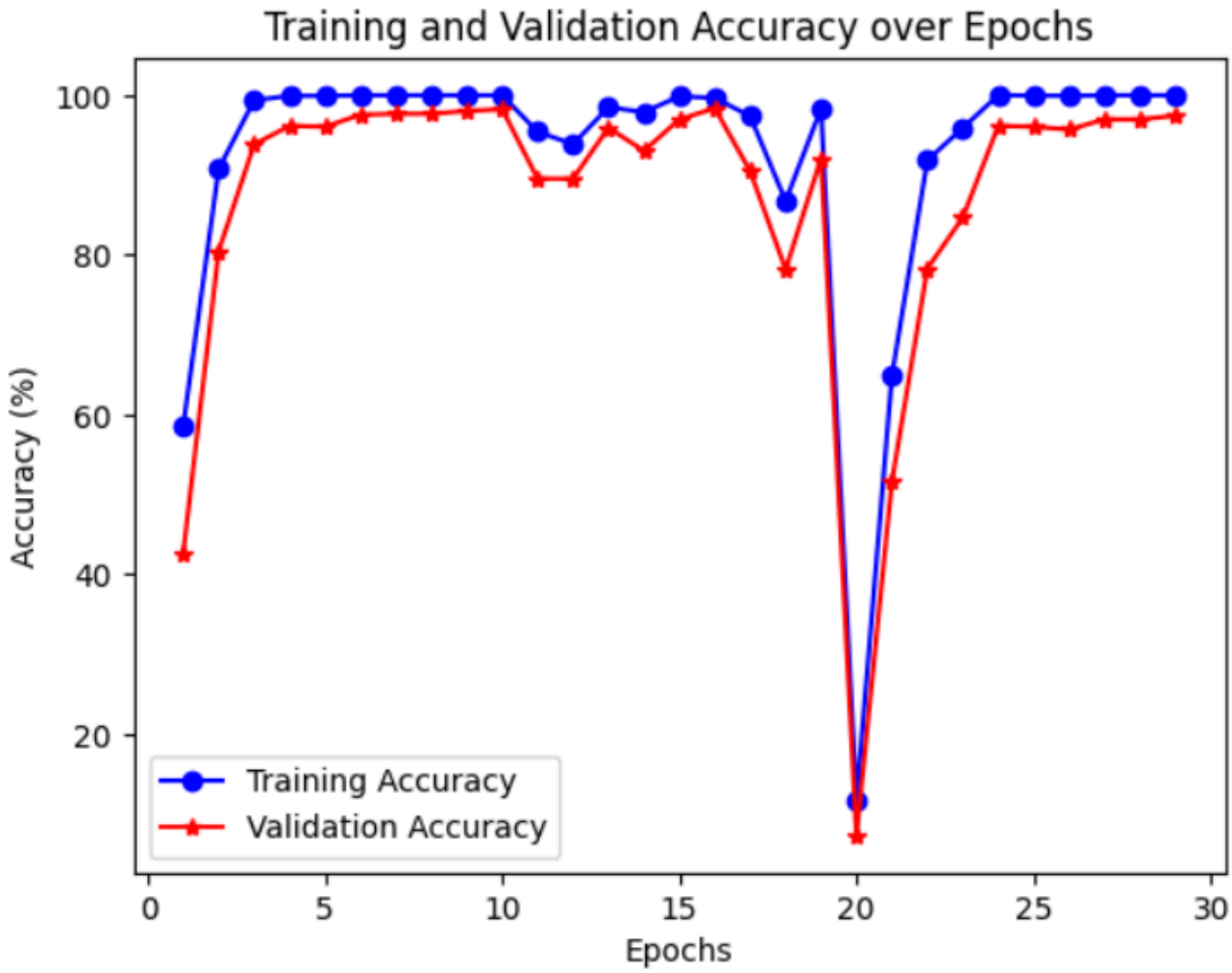
**ResNet18 as base model:**

Images are resized to 224x224 pixels, converted to PyTorch tensors, and normalized with ImageNet means and standard deviations.

**Train Dataset**: 70% of the total dataset is used for training.

**Validation Dataset**: 20% of the dataset is used for validation.

**Test Dataset**: 10% of the dataset is used for testing.




- **Training Accuracy (blue line)**:
  - The model's training accuracy increases rapidly in the first few epochs and then stabilizes close to 100%.
  - The overall performance is good, showing near-perfect accuracy.
  - A dip occurs around epoch 15, but the training accuracy quickly recovers, indicating some instability, possibly due to fluctuations in the learning process.
- **Validation Accuracy (red line)**:

- The validation accuracy follows a similar pattern to the training accuracy, increasing quickly and stabilizing around 95% after a few epochs.
- The occasional dips in validation accuracy (most notably around epoch 15 and 18) suggest some overfitting.
- Despite these dips, the validation accuracy stays reasonably high, indicating the model generalizes well overall.

Conclusion:

- **Good overall performance.**
- **Minor fluctuations.**
- **Slight overfitting.**

The model appears to perform well.

The model taking about 40 to 60 mins for an epoch.

**Siamese Network with DenseNet Backbone:**

We experimented with a Siamese Network using DenseNet as the base feature extractor.

The results were promising, showing better performance compared to a ResNet-based Siamese network. However, the results were difficult to compare iteratively due to the randomness in the generation of negative pairs. As a result, the variation across training epochs made it challenging to maintain consistent improvement. This approach would benefit from fine-tuning using a GPU for higher epochs, allowing us to explore the network's stability over a more extended training period.

**Siamese Network with ResNet Backbone:**

A Siamese Network with ResNet as the base model was explored. ResNet's deep architecture helps in feature extraction, but similar to the DenseNet version, the iterative updates led to high variance in results because of the random negative pair generation.

We observed that while DenseNet showed better overall performance, it is still necessary to conduct more controlled experiments using GPUs over more epochs to accurately assess the models' strengths and convergence properties.

**DCGAN + ResNet:**

We attempted a DCGAN (Deep Convolutional Generative Adversarial Network) to generate synthetic images for augmenting our training data, followed by using a ResNet-based

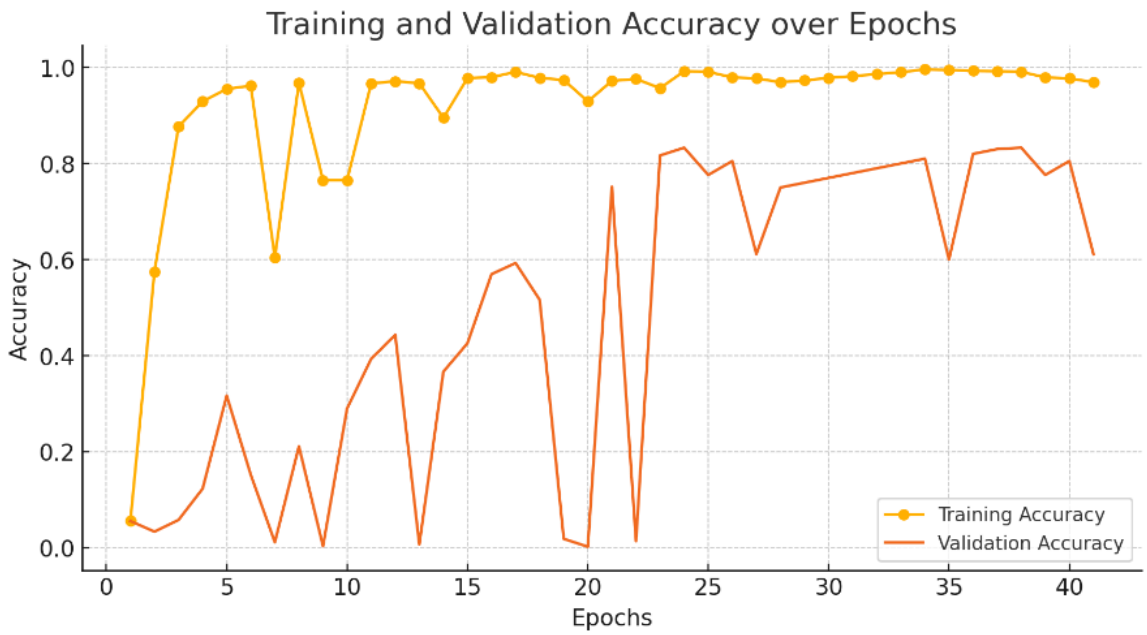
classifier. While DCGANs have the potential to increase the diversity of the training set,

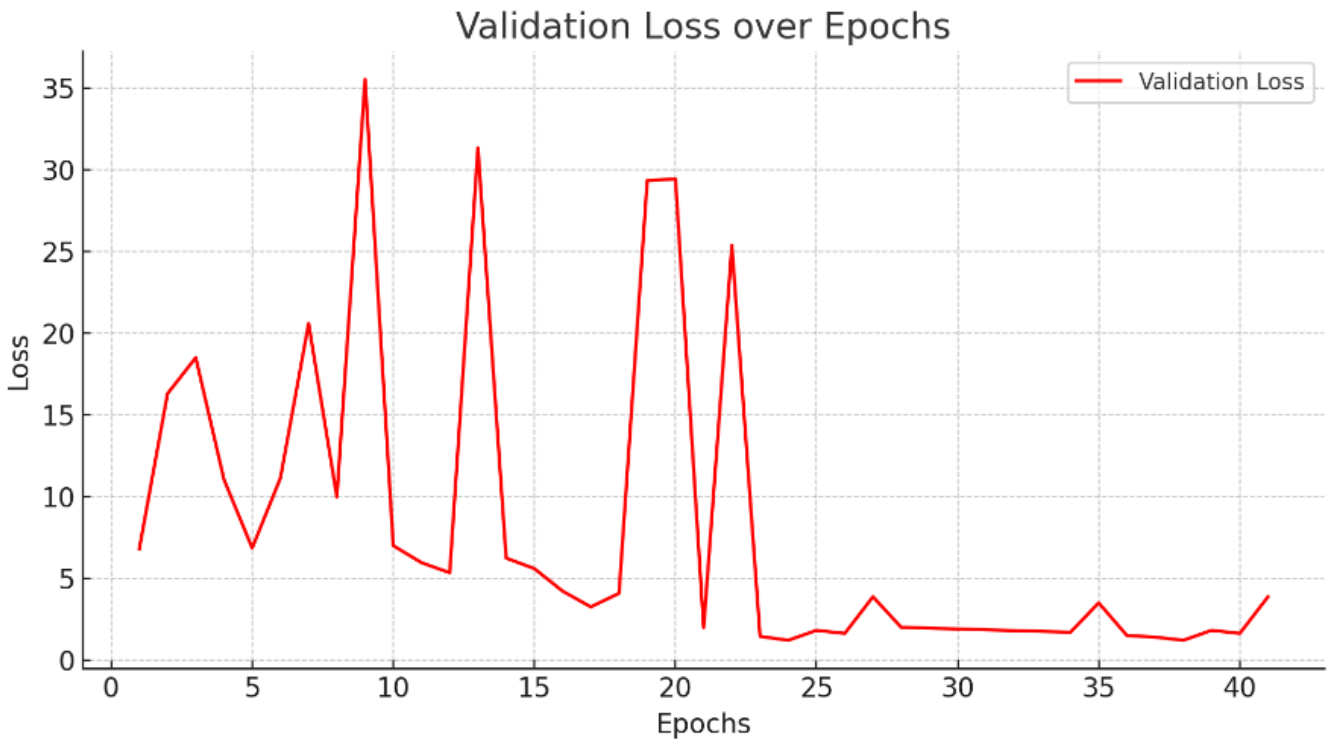
the implementation exceeded resource limits even for a single epoch due to the extensive computation required for both the generator and discriminator networks. This approach would need a more powerful setup to evaluate its utility fully.

**MobileNetV2 Backbone:**

We experimented with MobileNetV2 as a lightweight yet powerful feature extractor.

gave promising results to train compared to other architectures. We successfully ran it for 40 epochs until it crashed due to system limitations. Both training and validation accuracy stabilized around 30+ epochs, indicating good convergence.






This model has shown potential for this problem, and with sufficient resources, we can extend it to higher epochs to evaluate its performance comprehensively.