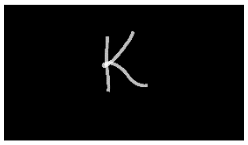
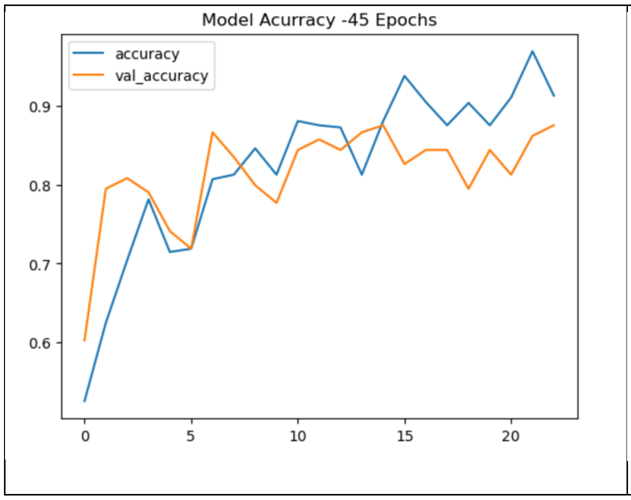
MNIST - CNN - My Handwritten Numbers

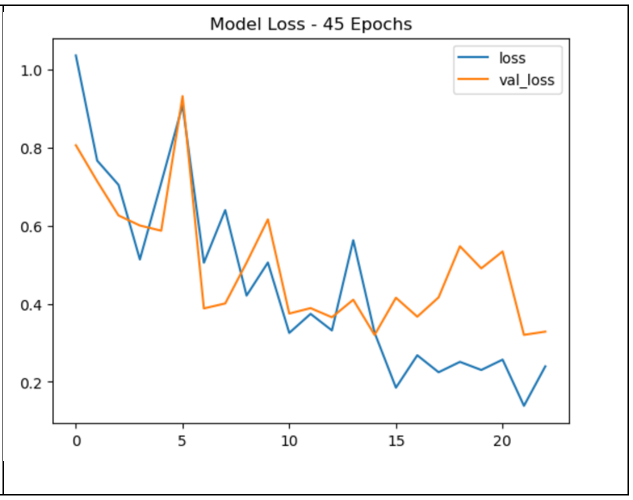
0	1
2	3
4	5
6	7
8	9
	

Model Accuracy: accuracy: **0.9903** - loss: **0.0323**

Notes: With nearly 100% accuracy, the model correctly identifies my handwritten numbers, while 13, K, and G (not in the table above but in the script) were recognized as numbers 0-9. It seems the model identified each new character input as a number with the most similar appearance from the image dataset. For example, 13 was identified as 9. (assuming not to replace the entire mnist dataset with my handwritten numbers and only test them as new data inputs in the 'testing with new data' section)

Visual Weather Systems - CNN





Ending hyperparameters used:

```
epochs = 45  # Increase epochs
dropout_rate = 0.5  # Adjust dropout rate
learning_rate = 0.001  # Adjust learning rate
```

- **Accuracy (Training)**: The model has an accuracy of about **91.25%** on the training dataset, which is high. This means that the model correctly predicted the weather conditions in approximately 91.25% of the classes it was trained on.
- **Validation Accuracy**: The validation accuracy is lower, at about **87.5%**. Basically, when the model is tested on new, unseen data (the validation set), its performance drops slightly, which is common since models tend to perform better on data they've seen before.
- **Loss (Training)**: The loss value is **0.239**, which is relatively low, which means that the model's predictions are close to the actual values in the training set.
- **Validation Loss**: The validation loss is **0.328**, which is higher than the training loss. This is expected as the model may not generalize as well to unseen data, but it's still a reasonable value.

Model Accuracy:
The accuracy plot represents the change in accuracy and validation accuracy over 45 epochs of training. The blue line (accuracy) remains stable around 0.95, while the orange line (validation accuracy) shows more fluctuation, showing a lot of variability in the model's performance on the validation data across different epochs.

Note that the model seems to be performing well, but there might be some overfitting to the training data since the validation accuracy is lower than the training accuracy.

Model Loss:

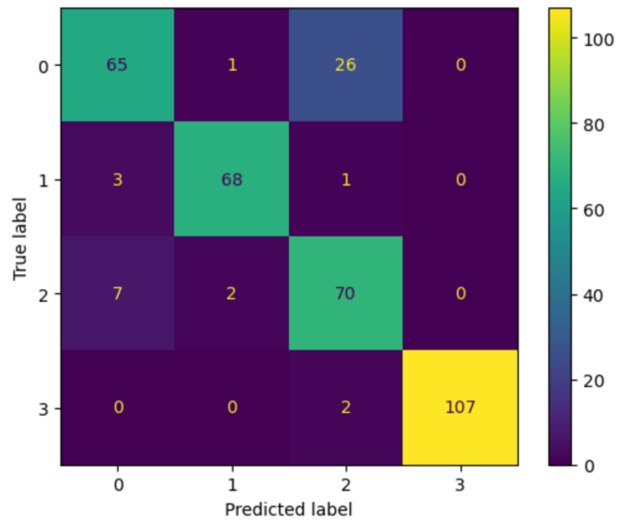
- **Learning**: The model is learning as shown by the decreasing training loss.
- **Overfitting**: The fluctuating validation loss suggests the model might be overfitting to the training data.
- **Generalization**: The higher validation loss compared to the training loss tells us that the model may not generalize as well to unseen data.
- **Epochs**: After a certain number of epochs, the model doesn't seem to improve much, which might be a good point to stop training to prevent overfitting.

Techniques like early stopping, regularization, or using more data might help improve the model's performance on unseen data.

Recommendations

- Monitor the validation loss to ensure the model is not overfitting.

Confusion Matrix Accuracy



Notes:

- **Cloudy (0)**: The model correctly predicted 'cloudy' 65 times but confused it with 'shine' 26 times and 'rain' once.
- **Rain (1)**: It accurately predicted 'rain' 68 times, but there were instances where 'cloudy' was mistaken for 'rain' 3 times, and 'shine' mistaken for 'rain' once.
- **Shine (2)**: 'Shine' was correctly identified 70 times, though 'cloudy' was incorrectly labeled as 'shine' 7 times, and 'rain' as 'shine' twice.
- **Sunrise (3)**: 'Sunrise' had the highest accuracy with 107 correct predictions, with a few misclassifications where 'shine' two times.

Weather IMG Predictions

```
Correct Prediction - class: Sunrise - predicted: Sunrise[2.6414605e-06 5.8856010e-08 2.5702020e-06 9.9999475e-01]
```

```
Correct Prediction - class: Rain - predicted: Rain[5.4165048e-07 9.9999738e-01 2.0755724e-06 7.1102388e-11]
```

```
Correct Prediction - class: Shine - predicted: Shine[3.6411046e-04 1.1191969e-03 9.9840885e-01 1.0794970e-04
```

Notes: All three weather image predictions were correctly identified as the correct weather class from our training sets..

GANs: Proposal and 3 Innovative Ideas Backed by Research

- Develop a model that detects weather conditions by audio classification.
    a. (current or historical sounds of weather conditions sampled from city, forest, suburb, and rural areas: thunder, rain, gusts of wind, hail, storms, calm, etc)
- Develop a model to give alerts in vehicles when entering bad weather conditions or when bad weather conditions are occurring or are going to occur. (air vehicles, road vehicles, major or local events)
    a. (more weather vehicle options?)
    b. (automatically adjust computer engine to ensure safer travel?) (pretty advanced stuff)
- Develop a model to determine optimal locations for new homes or vacation spots by environmental factors. (weather conditions, the frequency of natural disaster events)

Sources:

1.
    a. https://huggingface.co/learn/audio-course/chapter4/classification_models
    b. https://medium.com/@oluyaled/audio-classification-using-deep-learning-and-tensorflow-a-step-by-step-guide-5327467ee6ab
    c. https://paperswithcode.com/task/audio-classification
    d. https://www.analyticsvidhya.com/blog/2022/04/guide-to-audio-classification-using-deep-learning/
2.
    a. https://www.mdpi.com/1424-8220/24/2/522
    b. https://mdpi-res.com/d_attachment/ai/ai-03-00016/article_deploy/ai-03-00016.pdf?version=1650271516
    c. https://mdpi-res.com/d_attachment/sensors/sensors-20-05731/article_deploy/sensors-20-05731.pdf
    d. https://link.springer.com/article/10.1007/s11042-023-16453-z
3.
    a. https://propertymetrics.com/blog/location-physical-and-environmental-factors-in-real-estate/
    b. https://arxiv.org/pdf/2210.14266v1.pdf
    c. https://www.redfin.com/guides/climate-change-housing-impact/methodology