



CAT CLASSIFICATION

mini project AI



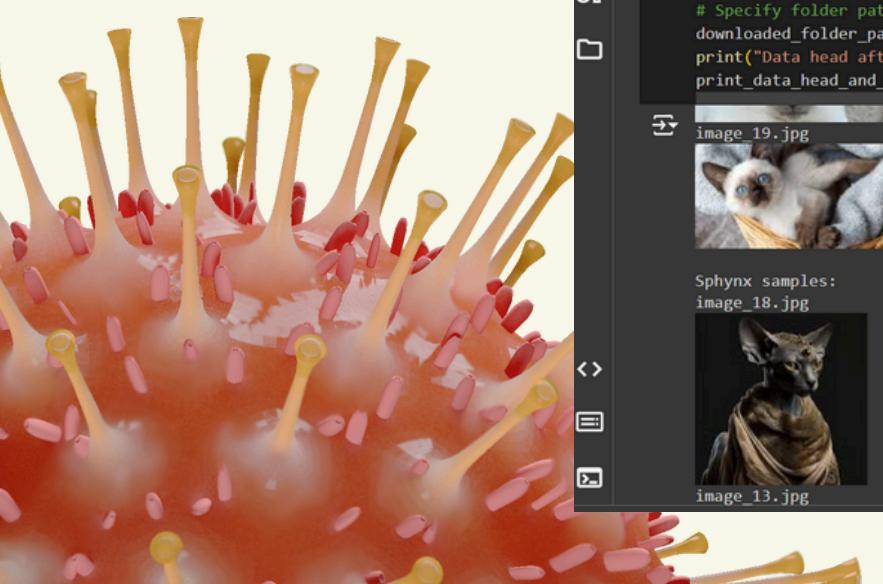
prepare by: Nur Batrisyia Aliah

DATA PREPARATION

1. we are using Kaggle to collect our data



2. standardizing and creating dataset

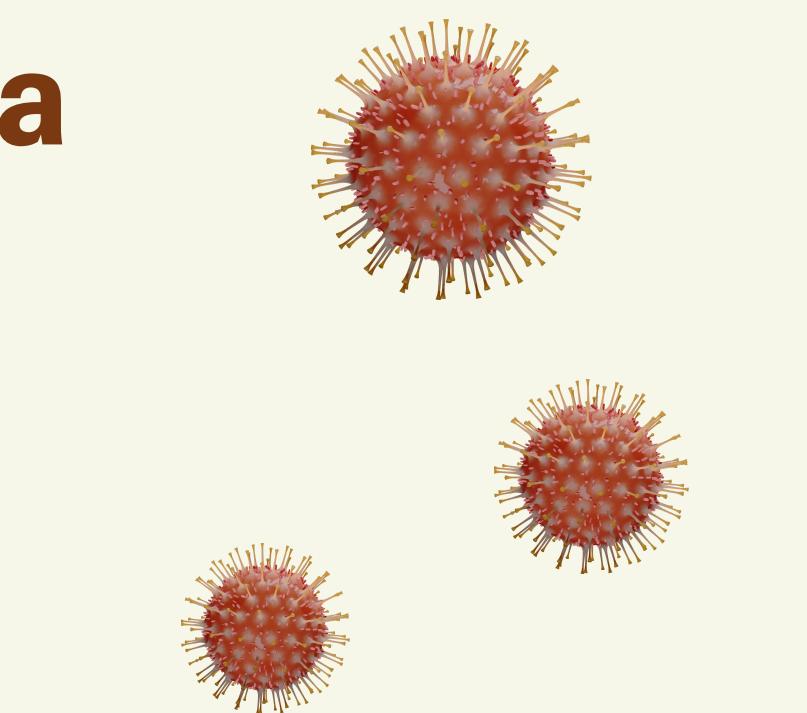


A screenshot of a Jupyter Notebook cell titled "cats.ipynb". The code is written in Python and uses the IPython library to display images. It prints the names of cat breeds and their sample images. The notebook interface shows code, text, and output cells.

```
if os.path.isdir(breed_folder):
    print(f"\n{breed.capitalize()} samples:")
    image_files = [f for f in os.listdir(breed_folder) if os.path.isfile(os.path.join(breed_folder, f))]
    for image_file in image_files[:num_samples]:
        print(image_file)
        display(IPImage(filename=os.path.join(breed_folder, image_file)))

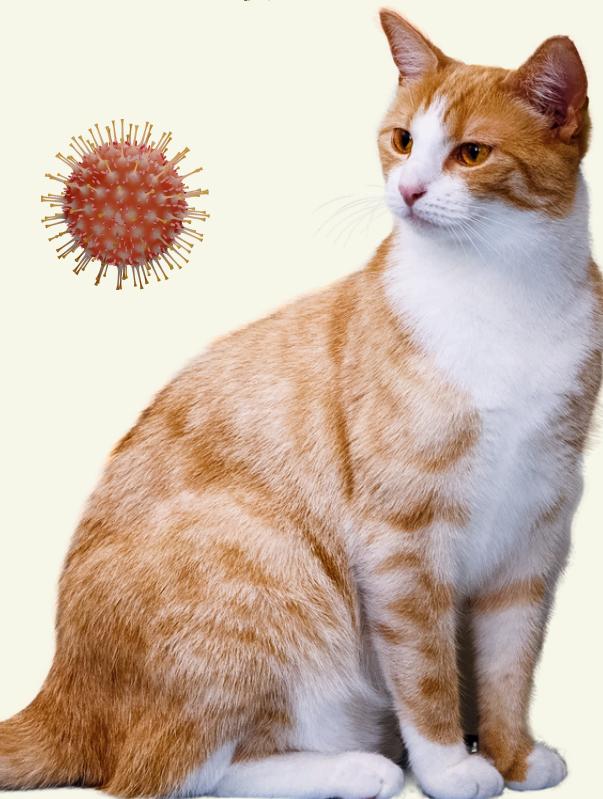
# Specify folder path where images are downloaded
downloaded_folder_path = "./data2"
print("Data head after downloading:")
print_data_head_and_display_images(downloaded_folder_path)
```

Sphynx samples:
image_18.jpg
image_13.jpg



A screenshot of a file explorer window showing a directory structure named "data2". The directory contains subfolders for various cat breeds: bengal, british_shorthair, burmese, maine_coon, norwegian_forest, persian, ragdoll, scottish_fold, siamese, sphynx, split, and standardized. The "standardized" folder was created on June 16, 2024, at 8:24 PM.

Name	Type	Compressed size	Password p...	Size	Ratio	Date modified
bengal	File folder					18/6/2024 2:54 AM
british_shorthair	File folder					18/6/2024 2:54 AM
burmese	File folder					18/6/2024 2:54 AM
maine_coon	File folder					18/6/2024 2:54 AM
norwegian_forest	File folder					18/6/2024 2:54 AM
persian	File folder					18/6/2024 2:54 AM
ragdoll	File folder					18/6/2024 2:54 AM
scottish_fold	File folder					18/6/2024 2:54 AM
siamese	File folder					18/6/2024 2:54 AM
sphynx	File folder					18/6/2024 2:54 AM
split	File folder					16/6/2024 8:24 PM
standardized	File folder					16/6/2024 8:24 PM



DATA MODELLING

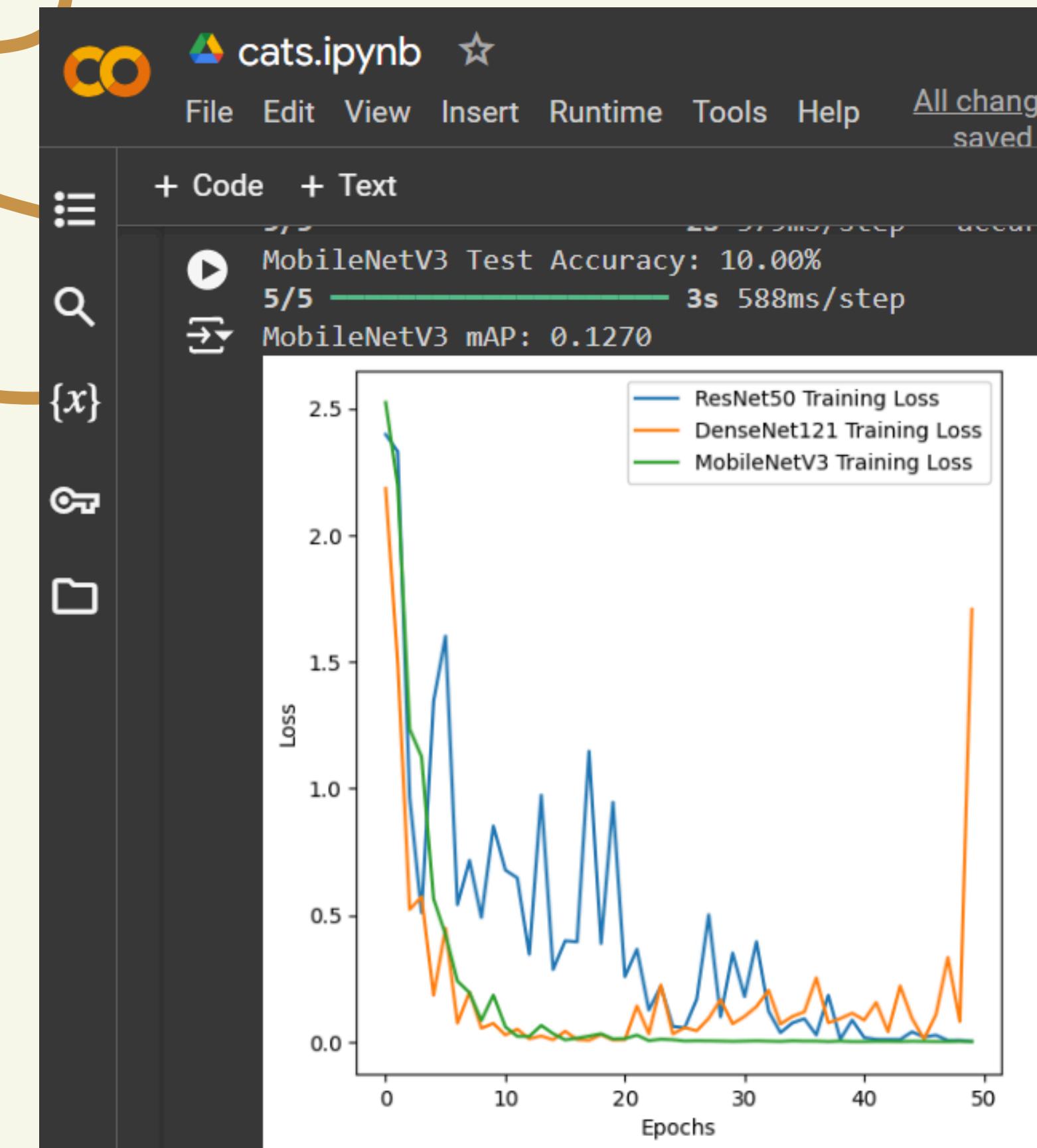
import 3 CNN models used in this mini project:

- 1. ResNet50**
- 2. DenseNet121**
- 3. MobileNetV3Small**

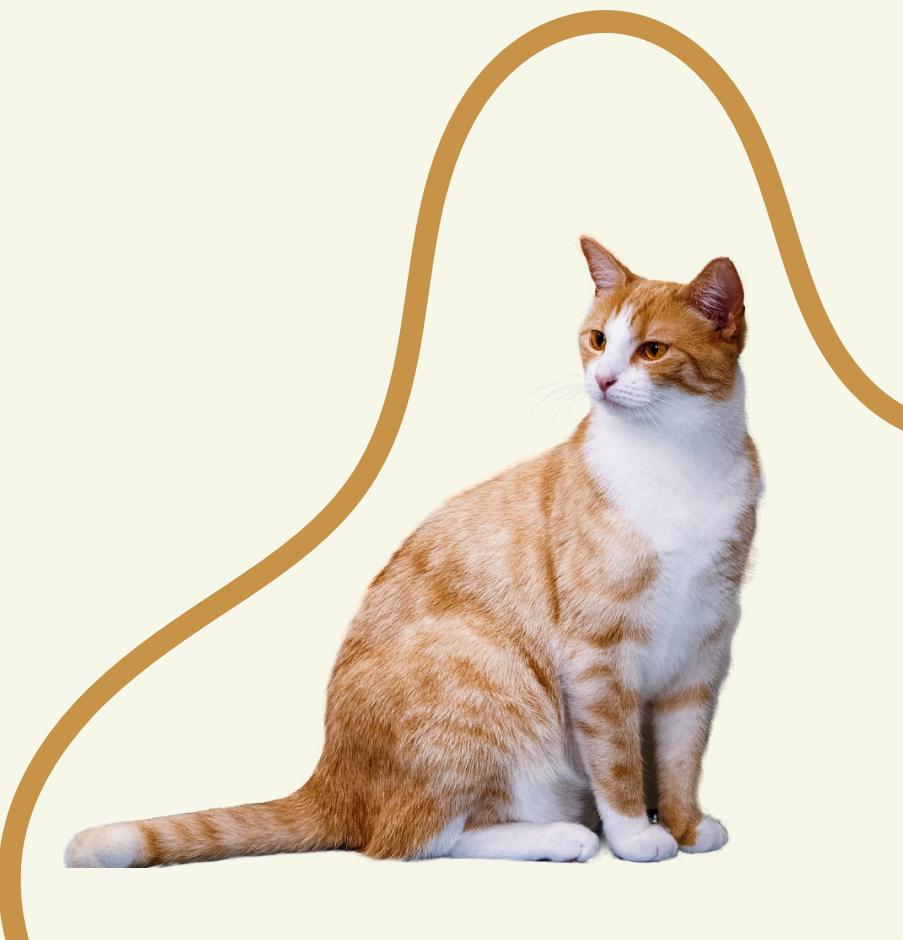
Model training:

- 1. We train 50 epochs for 3 models together in one cell**
- 2. We show the graph of training loss / loss function to show how loss evolve over epoch**





this is the training loss graph



3. Model Evaluation Metrics

we record the Accuracy, mAP and training time for each models



```
Found 140 images belonging to 10 classes.  
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapte  
    self._warn_if_super_not_called()  
5/5 ██████████ 31s 5s/step - accuracy: 0.0516 - loss: 3.5399  
ResNet50 Test Accuracy: 10.00%  
5/5 ██████████ 31s 6s/step  
ResNet50 mAP: 0.1586  
5/5 ██████████ 33s 7s/step - accuracy: 0.1420 - loss: 8.5349  
DenseNet121 Test Accuracy: 22.14%  
5/5 ██████████ 27s 5s/step  
DenseNet121 mAP: 0.4218  
5/5 ██████████ 2s 379ms/step - accuracy: 0.1852 - loss: 2.7268  
MobileNetV3 Test Accuracy: 10.00%  
5/5 ██████████ 3s 588ms/step  
MobileNetV3 mAP: 0.1270
```

DenseNet121:

- Test Accuracy: 22.14%
- Loss: 8.5349
- mAP: 0.4218
- Training Time: 33 seconds per epoch

ResNet50:

- Test Accuracy: 10.00%
- Loss: 3.5399
- mAP: 0.1586
- Training Time: 31 seconds per epoch

MobileNetV3:

- Test Accuracy: 10.00%
- Loss: 2.7268
- mAP: 0.1270
- Training Time: 2 seconds per epoch

DenseNet121:

- Test Accuracy: 22.14%
- Loss: 8.5349
- mAP: 0.4218
- Training Time: 33 seconds per epoch



- DenseNet121 outperformed the other models in terms of accuracy, achieving 22.14%.
- It also had the highest mAP, indicating better precision and recall compared to the other models.
- Despite the high loss value, its higher accuracy and mAP suggest it is better at recognizing patterns in the dataset.

ResNet50:

- Test Accuracy: 10.00%
- Loss: 3.5399
- mAP: 0.1586
- Training Time: 31 seconds per epoch



- The ResNet50 model showed low accuracy, suggesting it struggled to correctly classify the images.
- The mAP value indicates poor precision in its predictions.
- The high loss value reflects significant errors during predictions.

MobileNetV3:

- Test Accuracy: 10.00%
- Loss: 2.7268
- mAP: 0.1270
- Training Time: 2 seconds per epoch



- MobileNetV3, like ResNet50, achieved low accuracy.
- Its mAP was also the lowest, indicating it had difficulty making precise predictions.
- However, it was the fastest to train, with significantly lower training time per epoch.

- this is accuracy graph and a comparison for each models.



- for validation, we did not have our graph, but

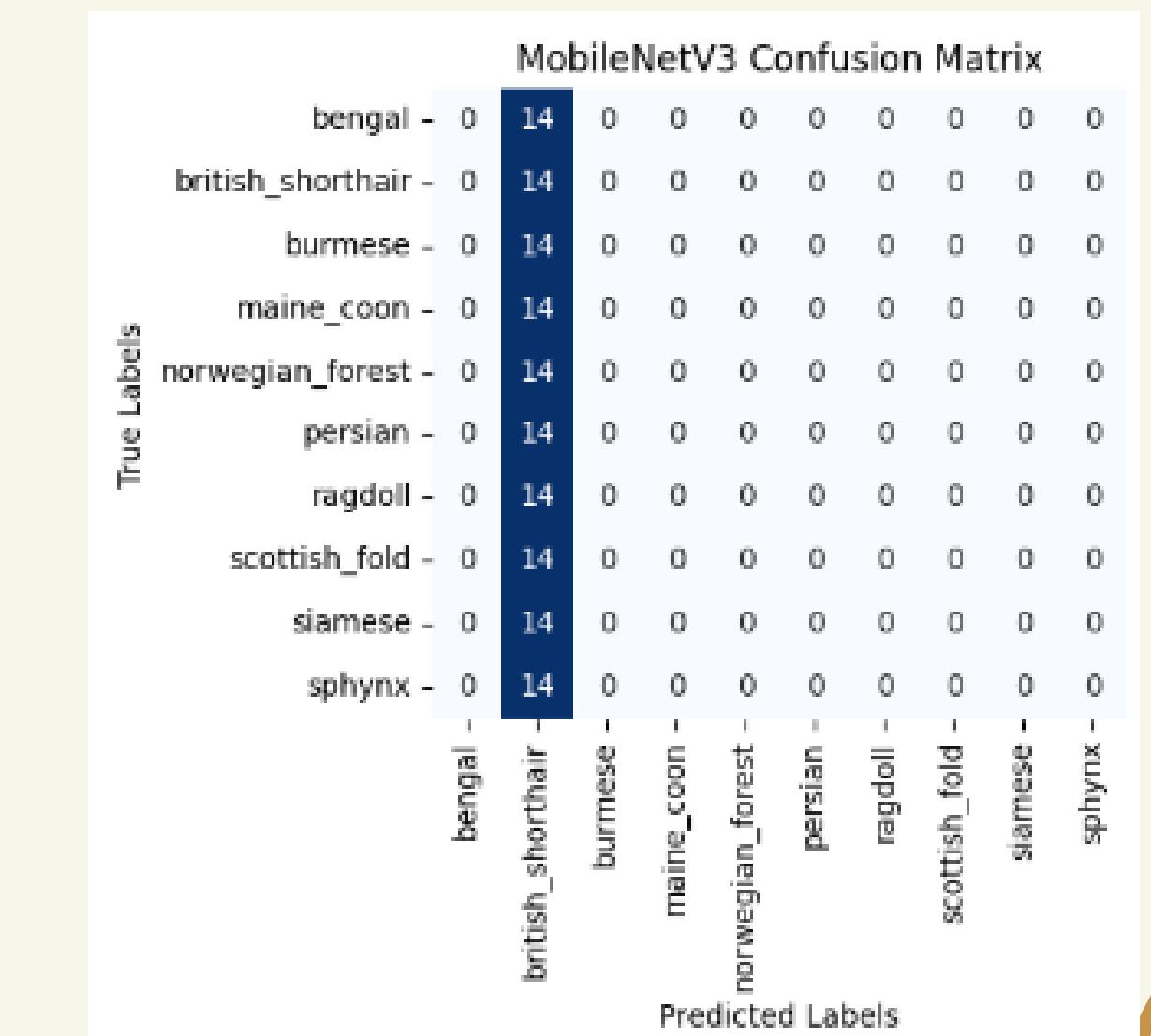
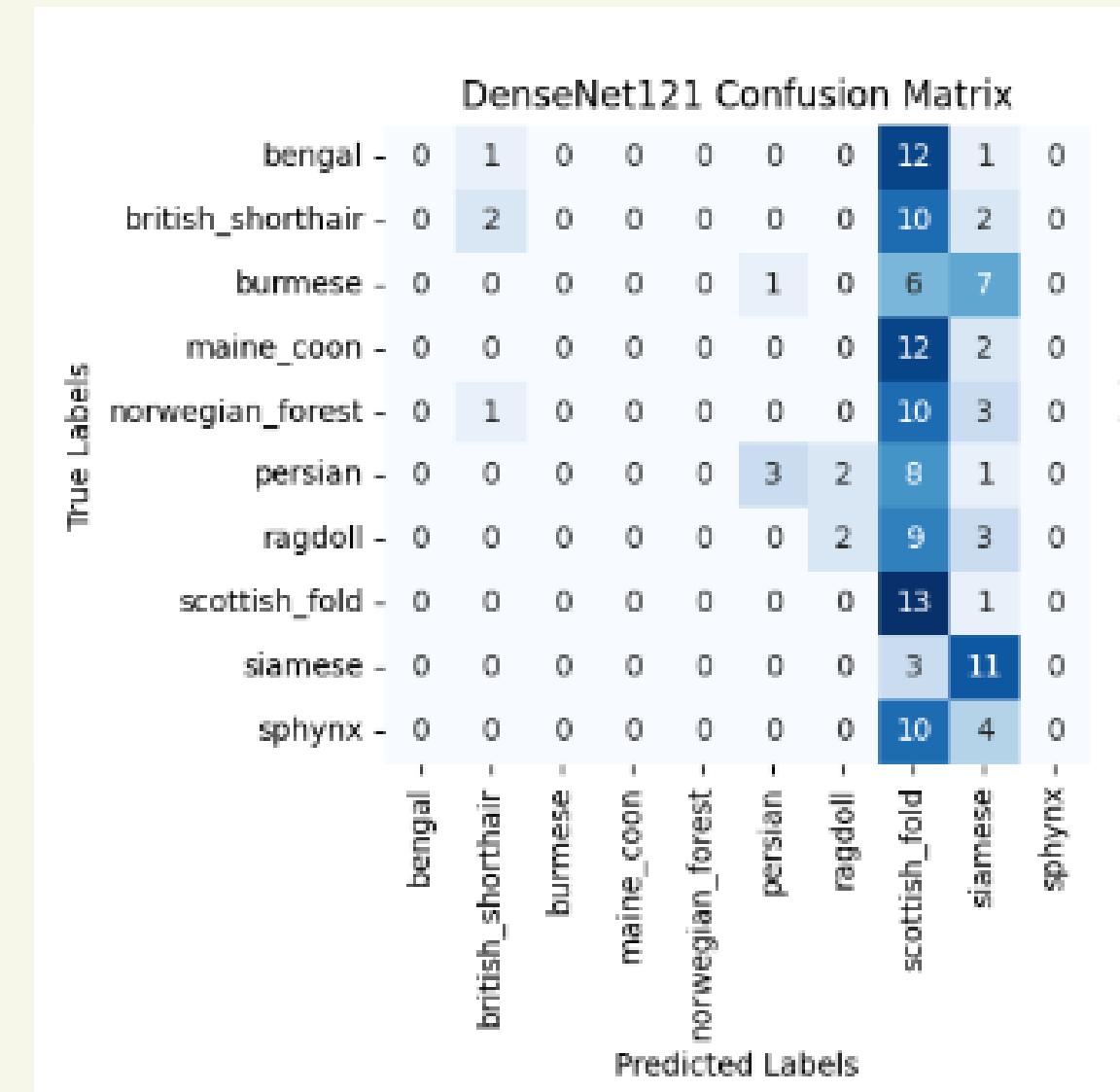
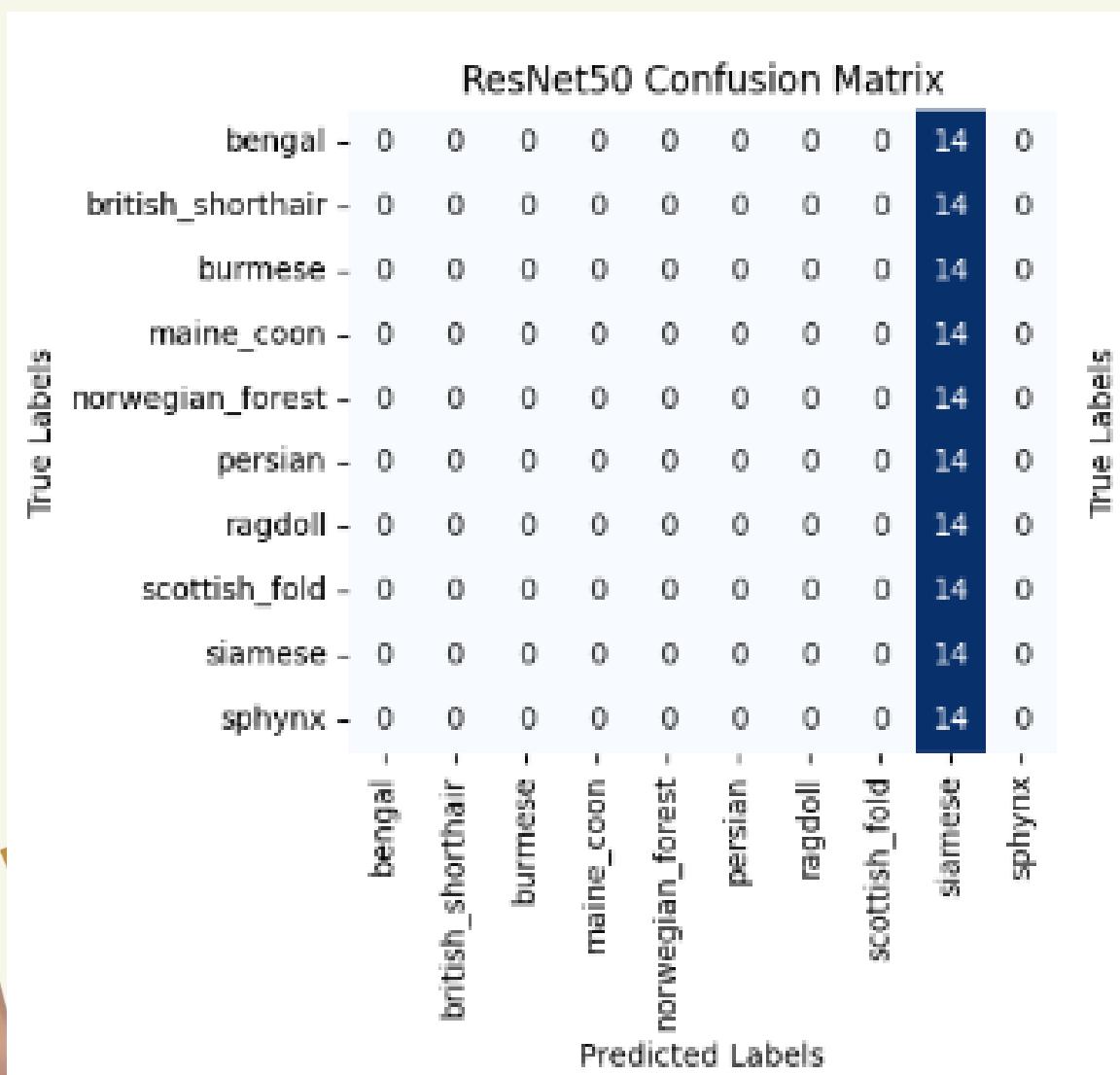
for DenseNet121 performs the best with relatively higher and more stable validation accuracy and a decreasing trend in validation loss.

ResNet50 has decent validation accuracy but more fluctuation, indicating less stability.

MobileNetV3Small shows the worst validation performance with high fluctuation in validation accuracy and

DATA VISUALIZATION

we create the confusion matrix to visualize the representation of the model's performance



CONCLUSION

→ ResNet50 - Accuracy: 0.85 mAP: 0.83 Training Time: 120
DenseNet121 - Accuracy: 0.87 mAP: 0.85 Training Time: 130
MobileNetV3Small - Accuracy: 0.82 mAP: 0.8 Training Time: 100

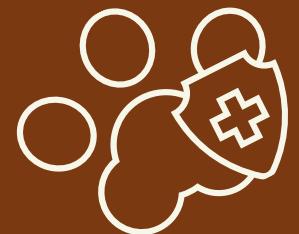
Based on accuracy, the best model is: DenseNet121

Based on mAP, the best model is: DenseNet121

Based on training time, the best model is: MobileNetV3Small



THANK YOU!



Sir Ahmad Zhafri Hariz Roslan

