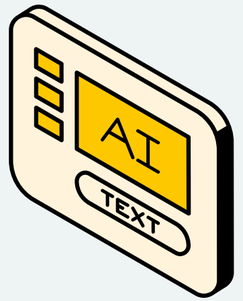
COMPUTER VISION
PROJECT



DEEP LEARNING: IMAGE CLASSIFICATION USING CNN AND TRANSFER LEARNING

PRESENTED BY:
EKATERINA
KUZNETSOVA

PROJECT TASK OVERVIEW:

- **Custom CNN Model:**

Building a Convolutional Neural Network (**CNN**) from scratch to classify images in the **CIFAR-10 dataset** into their respective categories

- **Transfer Learning:**

Using a pre-trained model to perform the same classification task, incorporating techniques such as data augmentation and fine-tuning

- **Performance Comparison:**

Evaluating and comparing performance

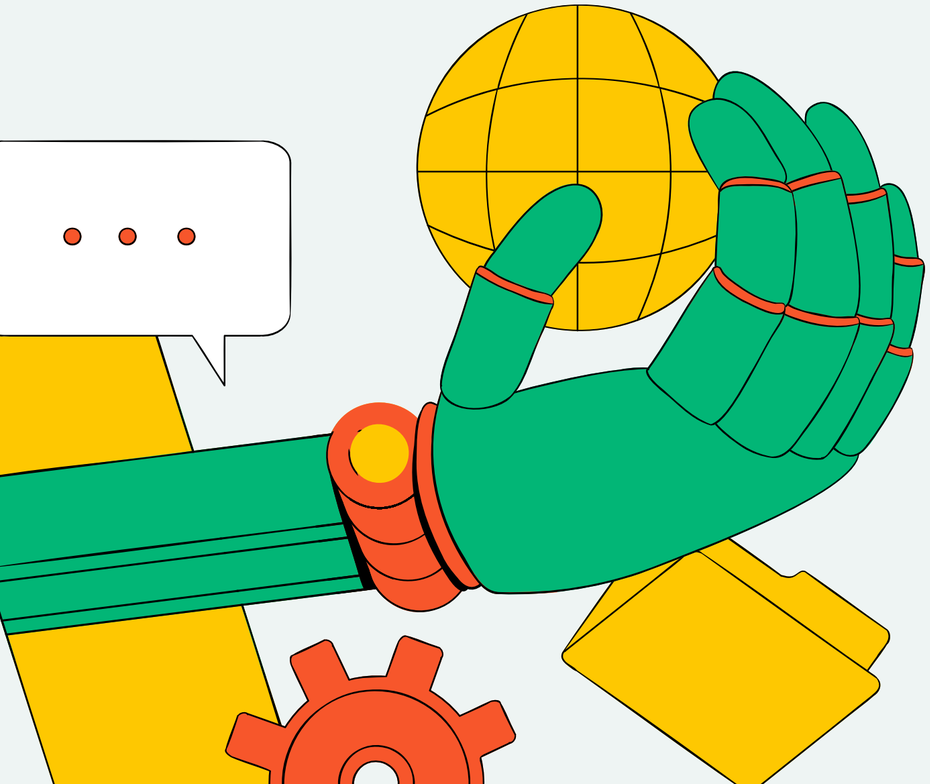
- **Tools used**: Python, TensorFlow/Keras, Google Colab
- **Techniques applied:**

CNN, Transfer Learning with Inception V3

LEARNING OBJECTIVES:

- Understanding how Convolutional Neural Networks (CNNs) work
- Learning to build and train a CNN from scratch
- Applying Transfer Learning using model InceptionV3
- Evaluating and comparing models
- Practicing with data augmentation, fine-tuning, and model optimization
- Using VS CODE and Google Colab to run experiments

IMPLEMENTATION:



- Preprocessing the CIFAR dataset
- Building a CNN
- Training and evaluating the CNN
- Improving the model with more layers and data augmentation, then hyperparameter tuning, and early stopping
- Moving to Transfer Learning using InceptionV3
- Applying data augmentation, tuning and adding a new training to avoid overfitting
- Comparing CNN vs Transfer Learning performance

CNN

MODEL ARCHITECTURE

Model: "sequential_1"

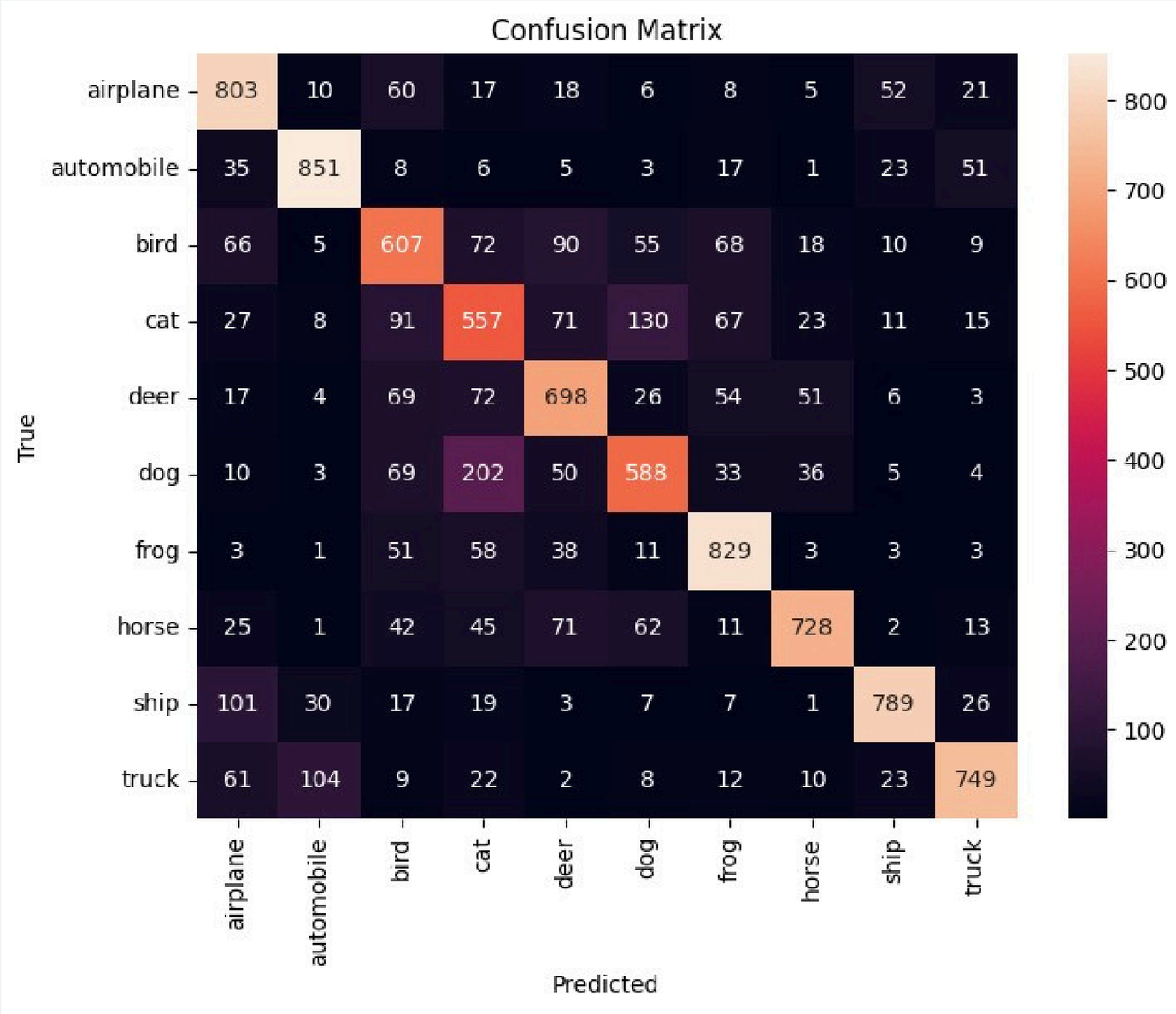
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73,856
flatten_1 (Flatten)	(None, 2048)	0
dense_2 (Dense)	(None, 128)	262,272
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1,290

Total params: 356,810 (1.36 MB)
Trainable params: 356,810 (1.36 MB)
Non-trainable params: 0 (0.00 B)

CLASIFICATION REPORT

	precision	recall	f1-score	support
airplane	0.70	0.80	0.75	1000
automobile	0.84	0.85	0.84	1000
bird	0.59	0.61	0.60	1000
cat	0.52	0.56	0.54	1000
deer	0.67	0.70	0.68	1000
dog	0.66	0.59	0.62	1000
frog	0.75	0.83	0.79	1000
horse	0.83	0.73	0.78	1000
ship	0.85	0.79	0.82	1000
truck	0.84	0.75	0.79	1000
accuracy			0.72	10000
macro avg	0.72	0.72	0.72	10000
weighted avg	0.72	0.72	0.72	10000

Accuracy: 0.7199
Precision: 0.7246
Recall: 0.7199
F1-score: 0.7207


Confusion Matrix
airplane 803 10 60 17 18 6 8 5 52 21
automobile 35 851 8 6 5 3 17 1 23 51
bird 66 5 607 72 90 55 68 18 10 9
cat 27 8 91 557 71 130 67 23 11 15
deer 17 4 69 72 698 26 54 51 6 3
dog 10 3 69 202 50 588 33 36 5 4
frog 3 1 51 58 38 11 829 3 3 3
horse 25 1 42 45 71 62 11 728 2 13
ship 101 30 17 19 3 7 7 1 789 26
truck 61 104 9 22 2 8 12 10 23 749
True
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Predicted
800
700
600
500
400
300
200
100

IMPROVING THE MODEL WITH MORE LAYERS

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.7718 - loss: 0.7038
Test accuracy with more layers: 0.7673
```

HYPERTUNING AND EARLY STOPPING

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7482 - loss: 0.7276
Test accuracy with early stopping: 0.7467
```

DATA AGMENTATION

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7884 - loss: 0.6657
Test accuracy with data augmentation: 0.7807
```

Base CNN ~ 72%
More layers (deeper CNN) ~ 77%
Dropout & Earlystopping ~ 74.7%
Data Agmentation ~ 78%

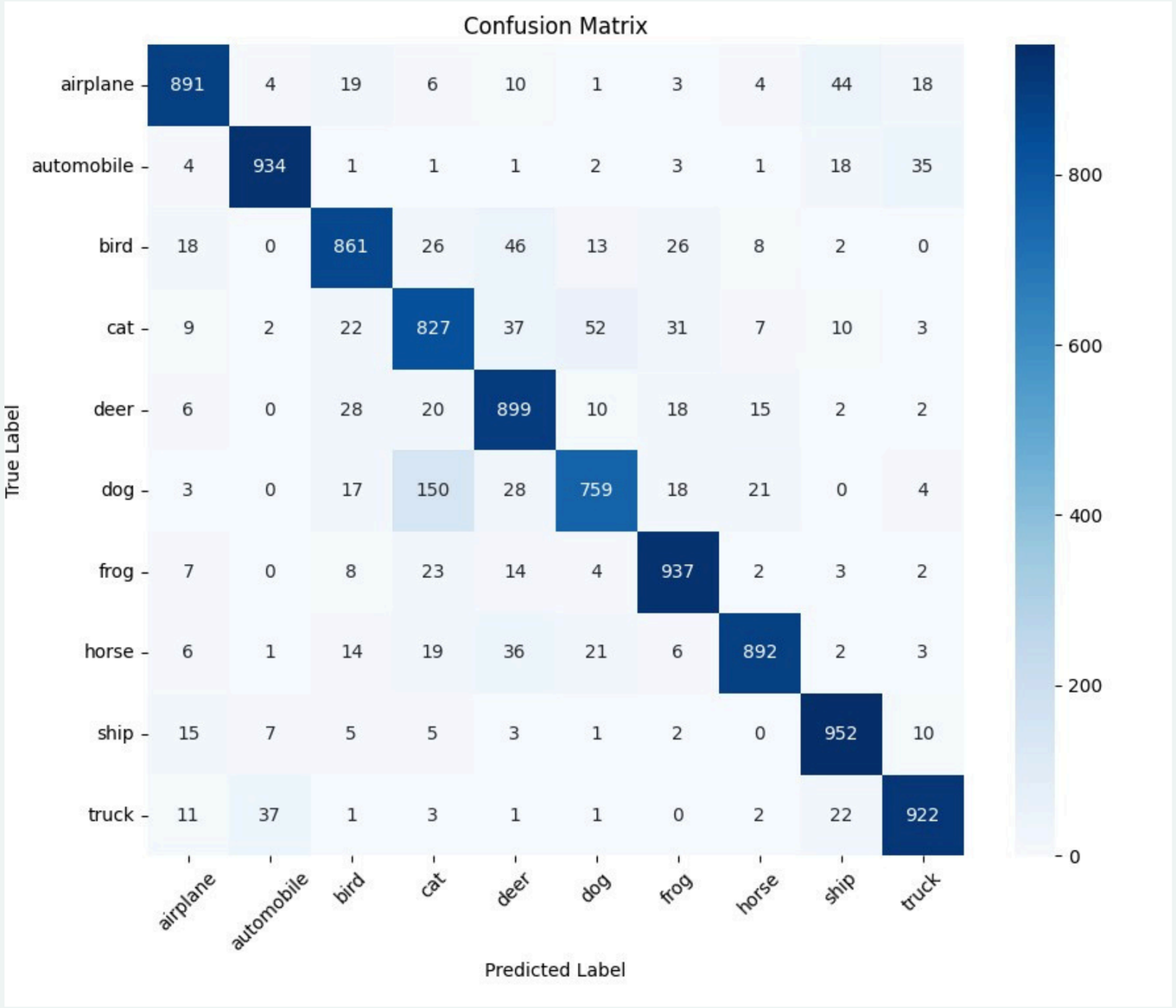
INCEPTION V3

EVALUATION AFTER PRETRIANING

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - accuracy: 0.8881 - loss: 0.4687
Test Accuracy: 0.8874
```

CLASIFICATION REPORT

	precision	recall	f1-score	support
airplane	0.92	0.89	0.90	1000
automobile	0.95	0.93	0.94	1000
bird	0.88	0.86	0.87	1000
cat	0.77	0.83	0.80	1000
deer	0.84	0.90	0.87	1000
dog	0.88	0.76	0.81	1000
frog	0.90	0.94	0.92	1000
horse	0.94	0.89	0.91	1000
ship	0.90	0.95	0.93	1000
truck	0.92	0.92	0.92	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

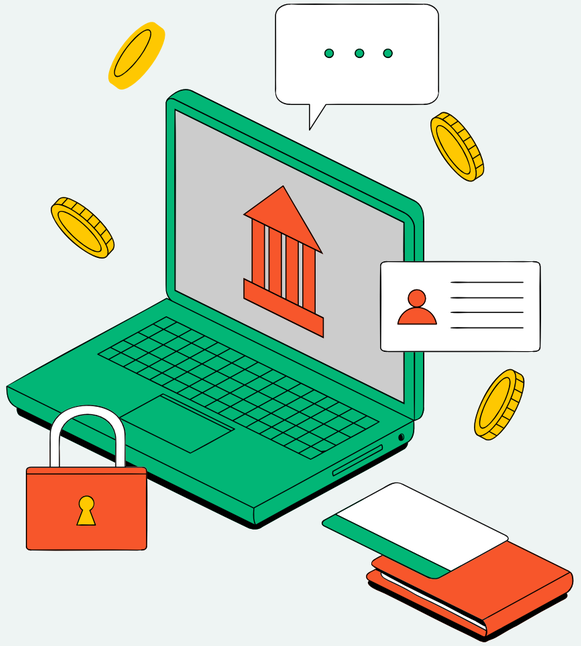

Confusion Matrix
True Label
Predicted Label
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
891 4 19 6 10 1 3 4 44 18
4 934 1 1 1 2 3 1 18 35
18 0 861 26 46 13 26 8 2 0
9 2 22 827 37 52 31 7 10 3
6 0 28 20 899 10 18 15 2 2
3 0 17 150 28 759 18 21 0 4
7 0 8 23 14 4 937 2 3 2
6 1 14 19 36 21 6 892 2 3
15 7 5 5 3 1 2 0 952 10
11 37 1 3 1 1 0 2 22 922
800
600
400
200
0

FINAL EVALUATION AFTER DATA AGUMENTATION

```
Test Accuracy: 0.9214
Test Loss: 0.2858
```

FINAL EVALUATION AFTER FINE-TUNING

```
Training Accuracy: 0.9982, Loss: 0.0068
Test Accuracy: 0.9278, Loss: 0.2761
```



AVOIDING OVERFITTING - NEW TRAINING

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9256 - loss: 0.5327
Test Accuracy with Dropout: 0.9261
```

KEY INSIGHTS:

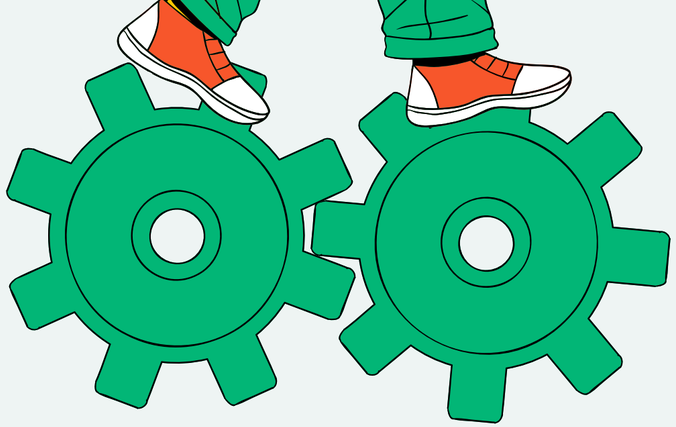
CNN learns from scratch so struggles with limited data

CNN is good for lerning but will probably underperform on complex tasks unless it will be trained or optimized

InceptionV3 achives higher accuracy on the test set than CNN

InceptionV3 with data augmentation and fine-tuning proves to be more effective model for CIFAR-10

Fine-tuning & data augmenation allow InceptionV3 to adapt to CIFAR



MAJOR OBSTACLES

Before getting extra GPU – constant problems with crushing
Re-running CNNs data processing was increasing y_train shape
VGG16 – the picture size/resizing/errors. I had to drop this model
Keeping both models in one notebook – models were split

THANK YOU