MULTI-CLASS ANIMAL IMAGE CLASSIFICATION

Creating a custom image classification dataset featuring five classes: dog, cow, cat, lamb, and zebra, each with 100 images sourced from the internet or captured using our phone.

Developing a classification model to classify these classes with at least 90% accuracy.

Mubashshira Kaisar
2221070642

Noshin Nawar
2221507042

Group No. - 1

Md Shihabul Alam Shihab
2233115642

Md. Thaminuzzaman
2312877642

IMPLEMENTATION USING CNN

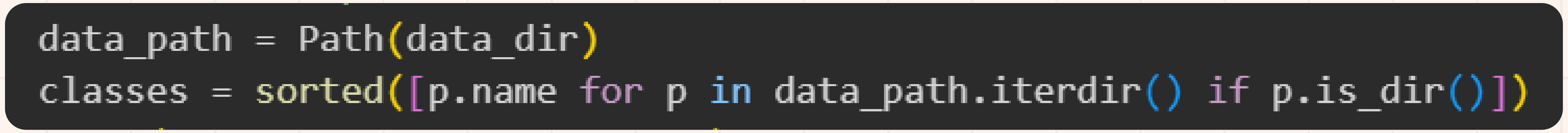
DATA PREPARATION & PROCESSING

Objective

Preparing an image dataset for training a deep learning model to classify animals.

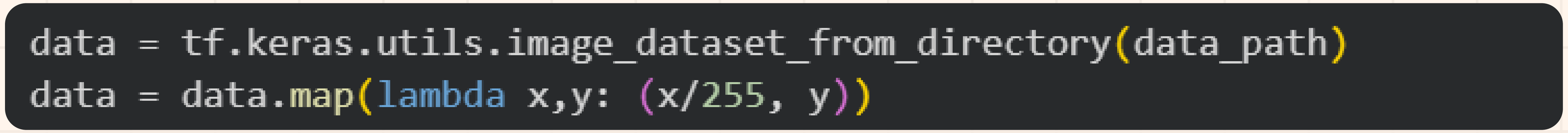
Dataset Setup

- Dataset contains **5 classes**: **cat, cow, dog, lamb, zebra**
- 100 images per class → total **500 images**
- Stored in class-wise folders (required by TensorFlow)



```
data_path = Path(data_dir)
classes = sorted([p.name for p in data_path.iterdir() if p.is_dir()])
```

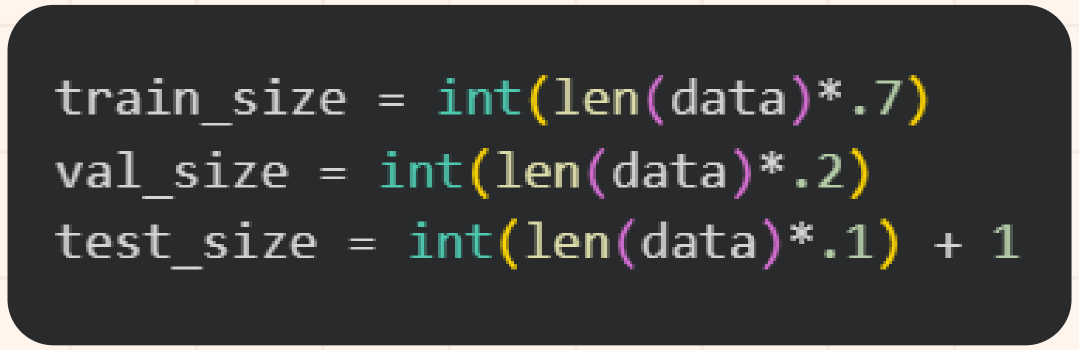
→ Automatically detects class labels from folder names



```
data = tf.keras.utils.image_dataset_from_directory(data_path)
data = data.map(lambda x,y: (x/255, y))
```

→

- Loads images
- Normalizes pixel values from **0–255 → 0–1**

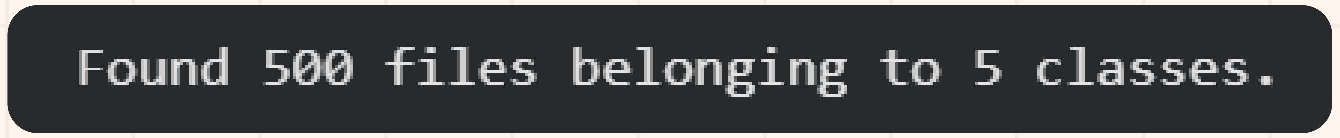


```
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1) + 1
```

Data Splitting

→

- **70% Training**
- **20% Validation**
- **10% Testing**



```
Found 500 files belonging to 5 classes.
```

Output

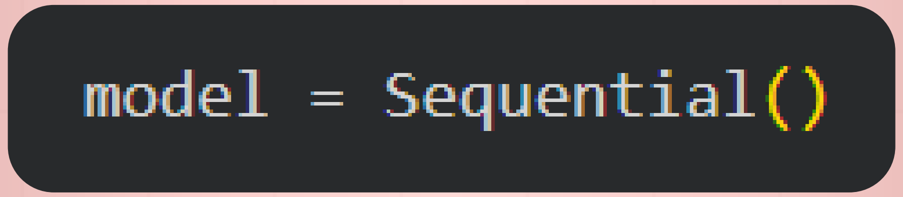
✅ Confirms dataset loaded correctly

CNN MODEL ARCHITECTURE

Why CNN?

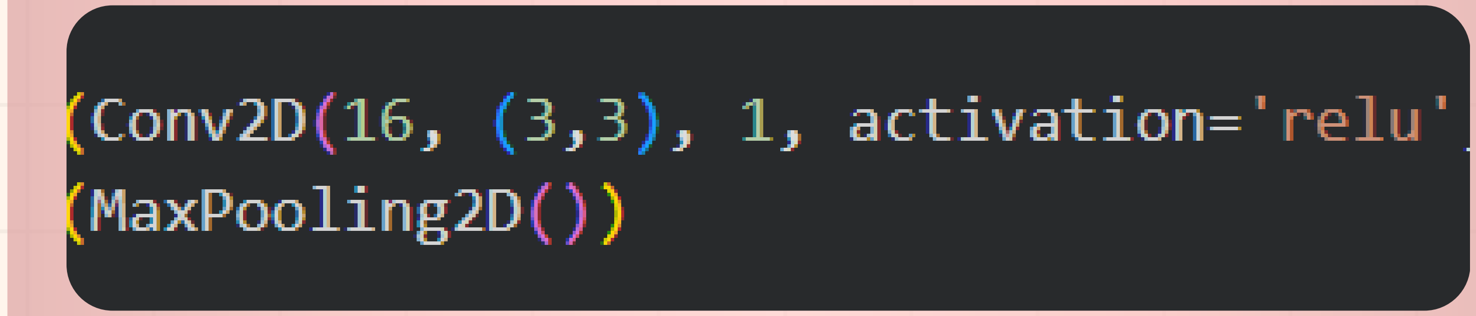
- Convolutional Neural Networks are effective for image feature extraction
- Automatically learn edges, textures, and shapes

Model Structure



```
model = Sequential()
```

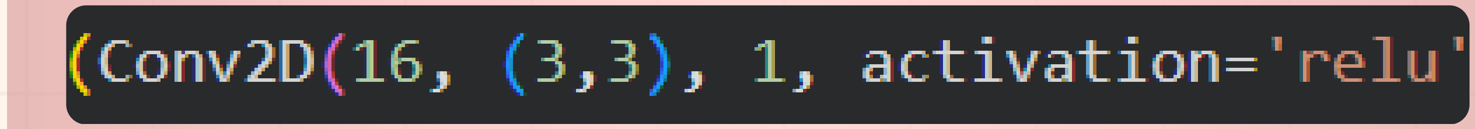
Convolution & Pooling Layers



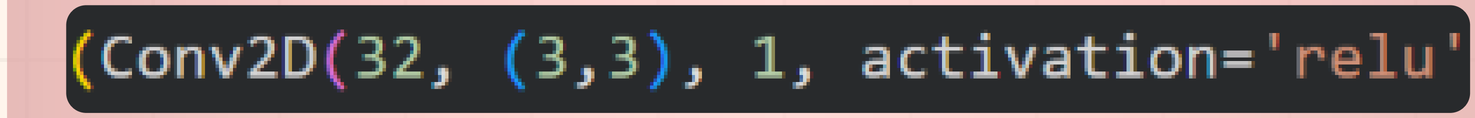
```
(Conv2D(16, (3,3), 1, activation='relu'
(MaxPooling2D())
```

→

- Extracts low-level features (edges)
- Pooling reduces image size & computation



```
(Conv2D(16, (3,3), 1, activation='relu'
```



```
(Conv2D(32, (3,3), 1, activation='relu'
```

→

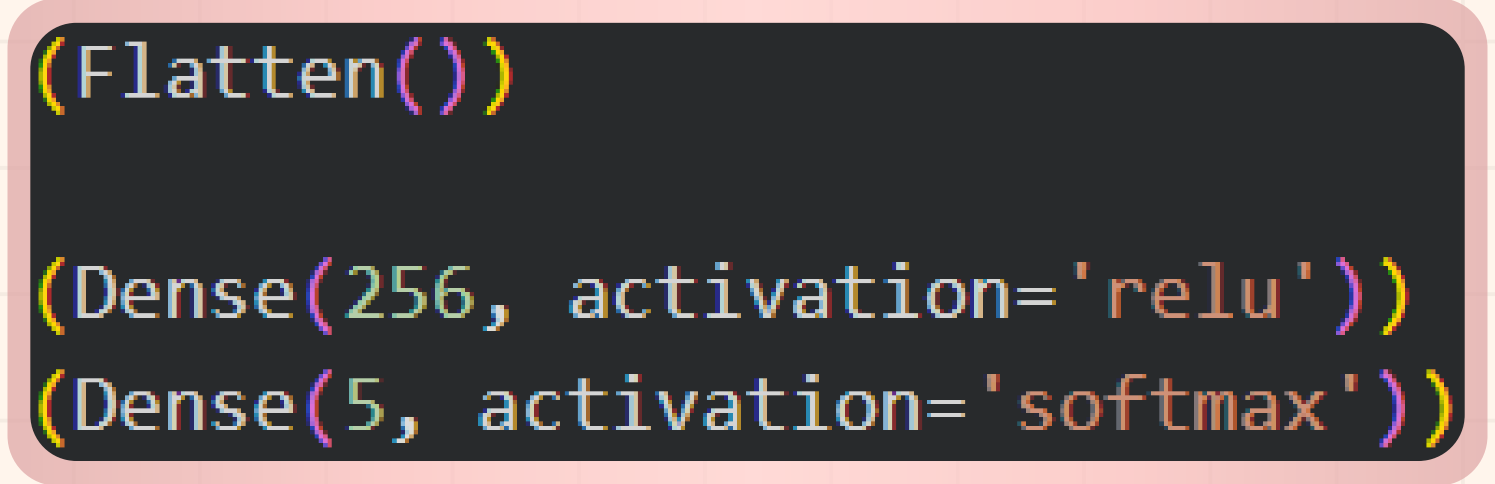
- Learns more complex patterns

CNN MODEL ARCHITECTURE (CONT.)

Fully Conne
-cted Layers

→

→



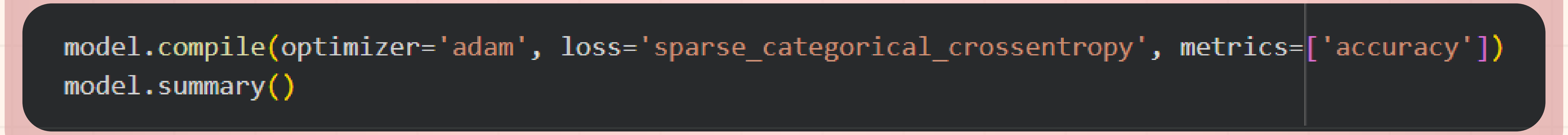
```
(Flatten())

(Dense(256, activation='relu'))
(Dense(5, activation='softmax'))
```

→

- Converts feature maps into vectors
- **Softmax outputs class probabilities (5 classes)**

Compilation



```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

→

- Adam optimizer
- Correct loss function for multi-class classification

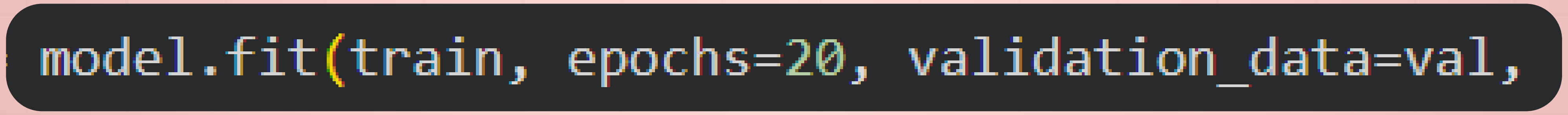
Output

3.69 million trainable parameters

Model summary confirms correct layer flow

TRAINING & PERFORMANCE ANALYSIS

Model Training



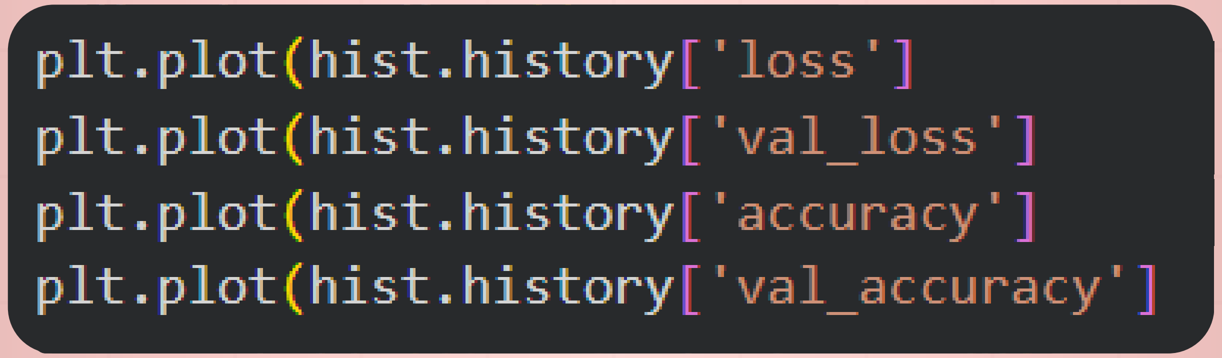
```
model.fit(train, epochs=20, validation_data=val,
```

- Trained for 20 epochs,
- Validation data monitors overfitting

Training Results

- Training accuracy → 100%
- Validation accuracy → 100%
- Loss → approaches 0

Loss & Accuracy Graph



```
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
```

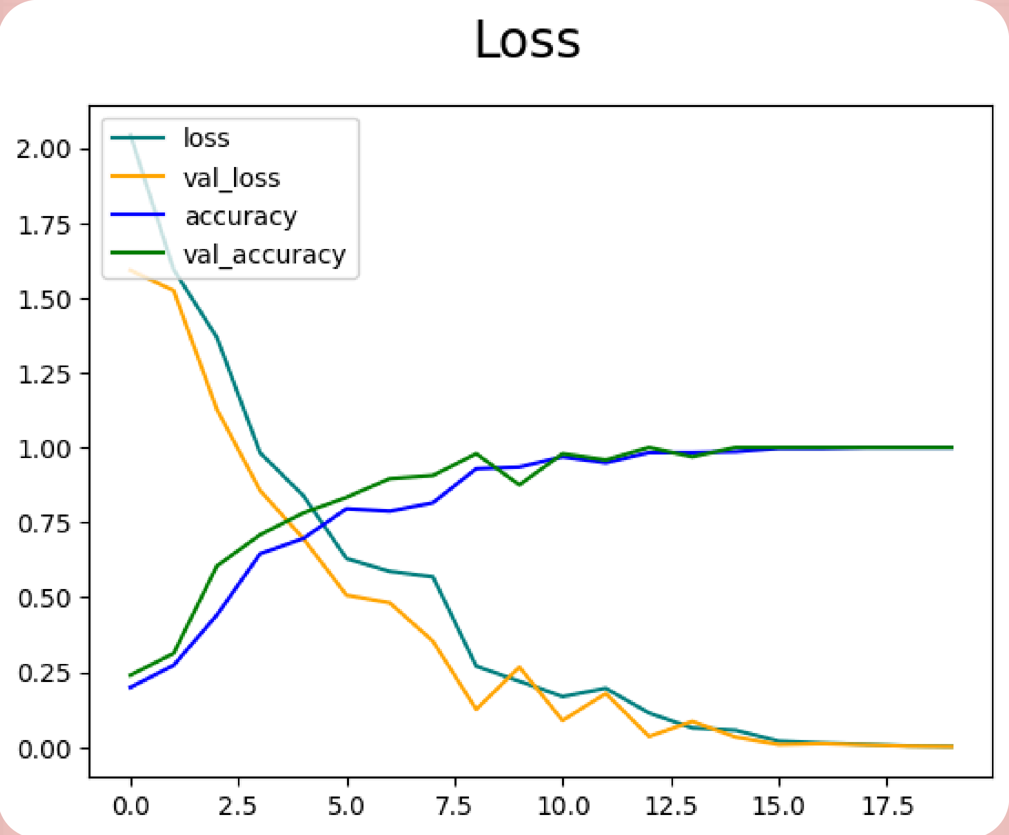
Key Observation:

✓ Model learned class features very effectively

✓ Dataset is clean and well-structured

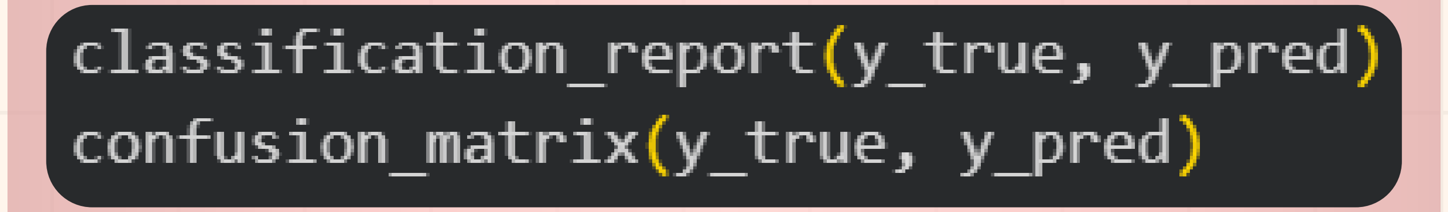
Graph Interpretation

- Loss decreases smoothly → good learning
- Validation loss closely follows training loss → no overfitting
- Accuracy increases steadily → model convergence



EVALUATION, TESTING & PREDICTION

Model Training



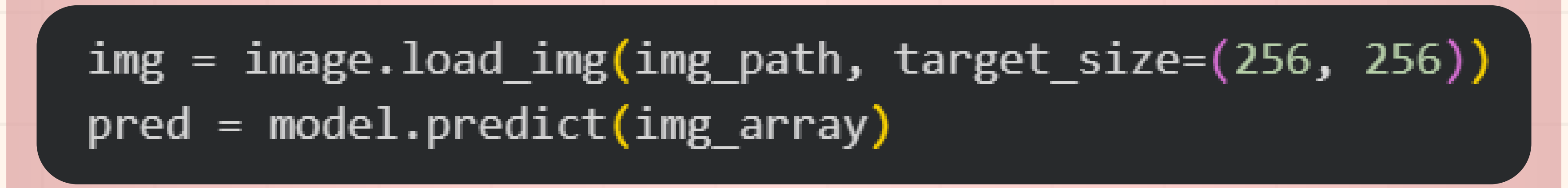
```
classification_report(y_true, y_pred)
confusion_matrix(y_true, y_pred)
```

Evaluation Output

- **Accuracy: 100%**
- **Precision, Recall, F1-score = 1.00 for all classes**
- **Confusion matrix shows zero misclassifications**

✓ Perfect classification on test set

Testing on New Image



```
img = image.load_img(img_path, target_size=(256, 256))
pred = model.predict(img_array)
```

Prediction Output

```
Predicted Class:", pred_class
```

- Uploaded image correctly classified
- Image displayed with predicted label

Final Conclusion →

- CNN successfully classifies animal images
- Proper preprocessing + balanced dataset = excellent performance
- Model is ready for real-world testing or deployment

Mubashshira Kaisar
2221070642

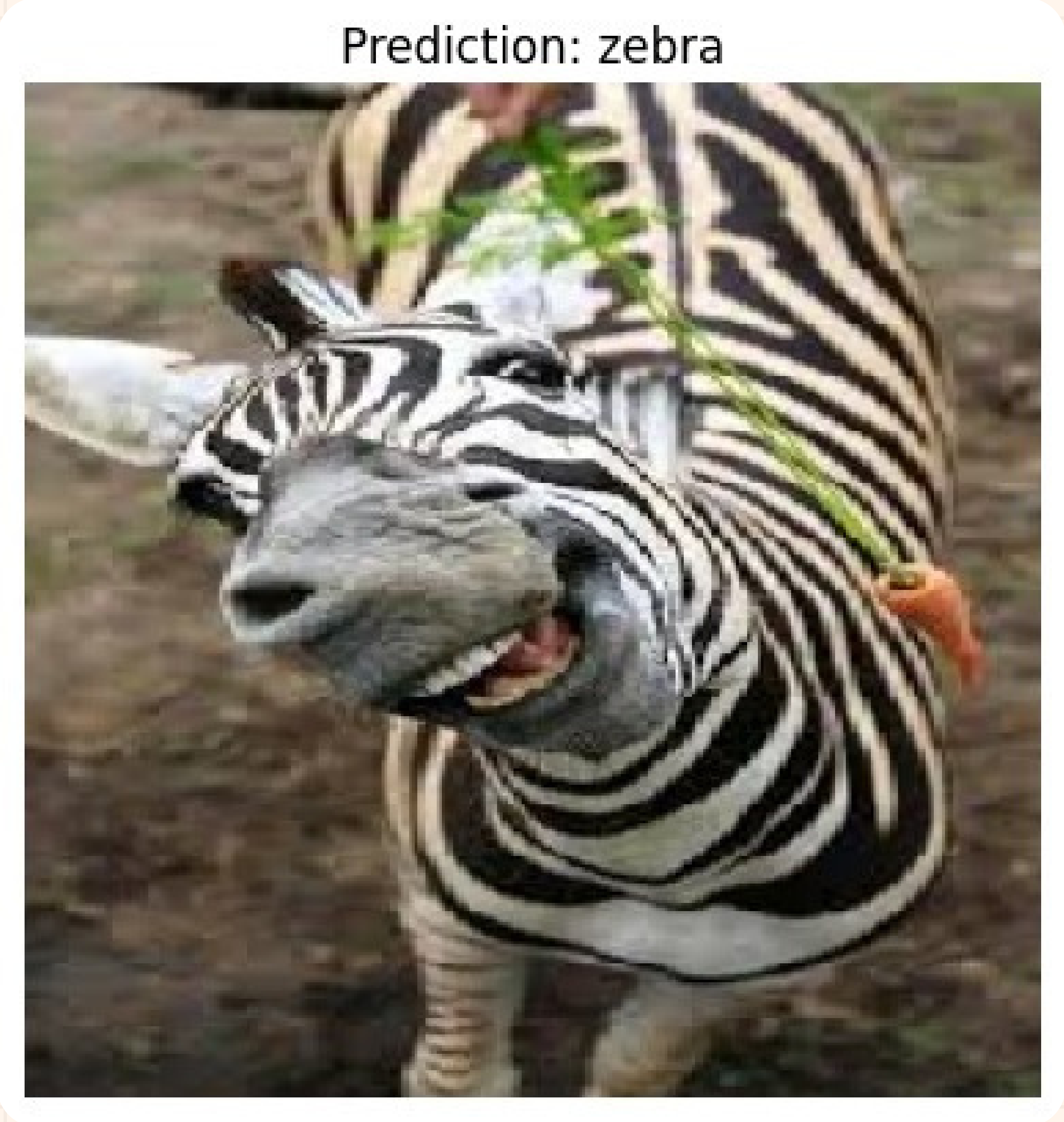
Noshin Nawar
2221507042

Group No. - 1

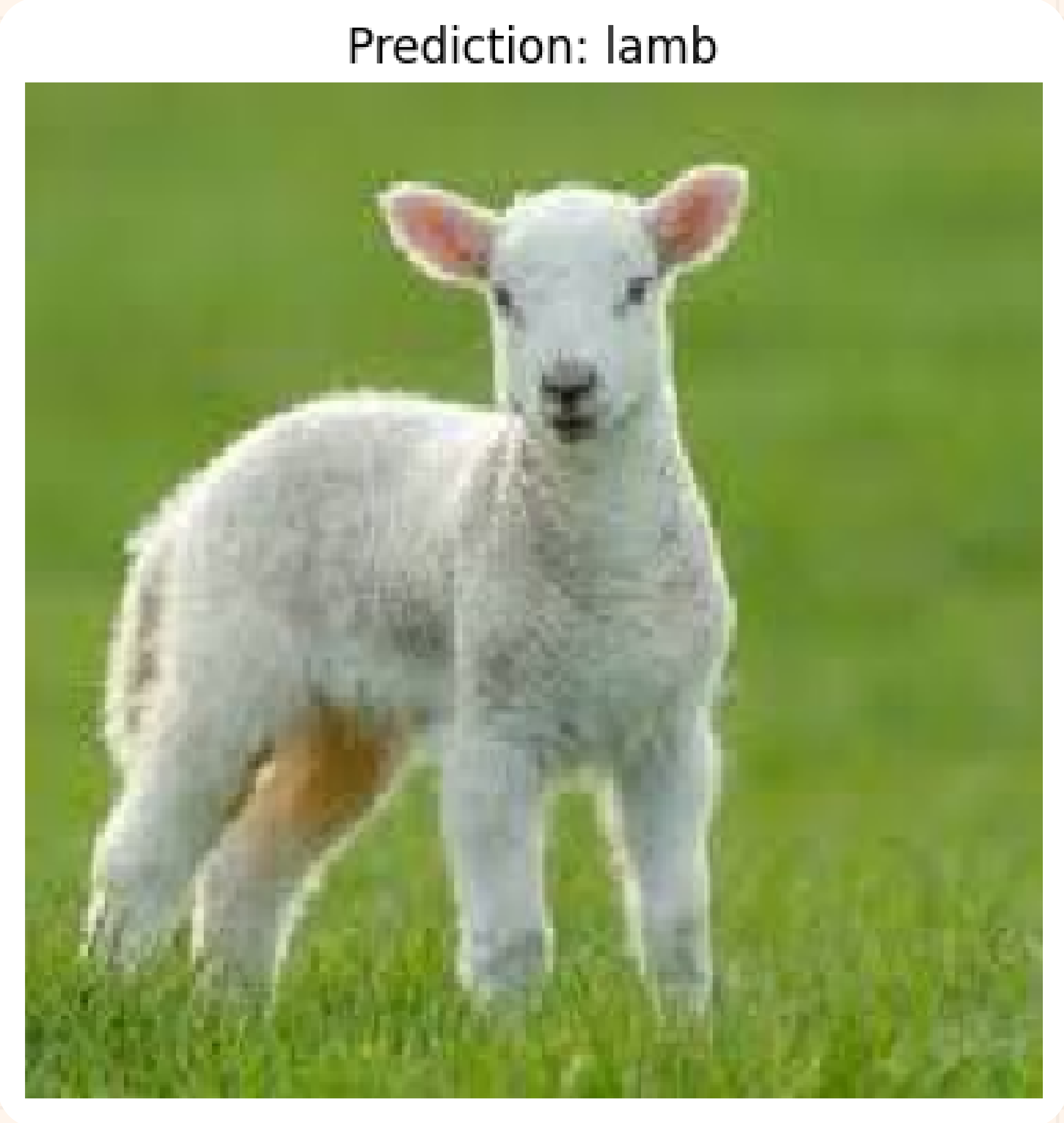
Md Shihabul Alam Shihab
2233115642

Md. Thaminuzzaman
2312877642

OUTPUT SCREENSHOTS



```
Prediction: zebra
```



```
Prediction: lamb
```

IMPLEMENTATION USING YOLOV11

Mubashshira Kaisar
2221070642

Noshin Nawar
2221507042

Group No. - 1

Md Shihabul Alam Shihab
2233115642

Md. Thaminuzzaman
2312877642

DATA PREPARATION & PROCESSING

Objective

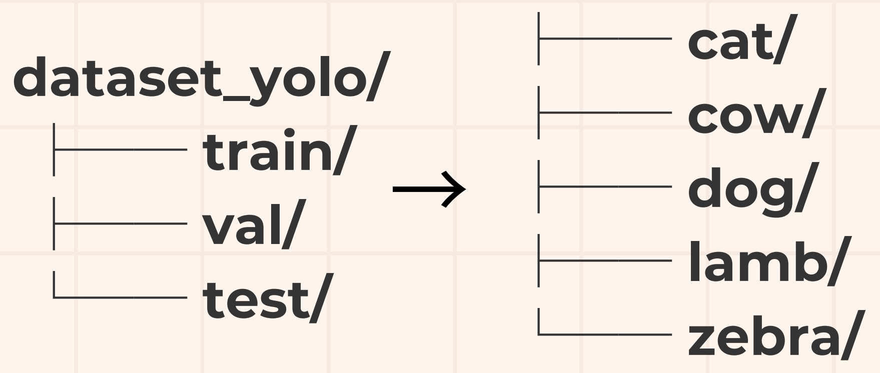
Preparing a raw image dataset stored in Google Drive into a YOLO-compatible classification dataset with train, validation, and test splits.

YOLO Classification expects:

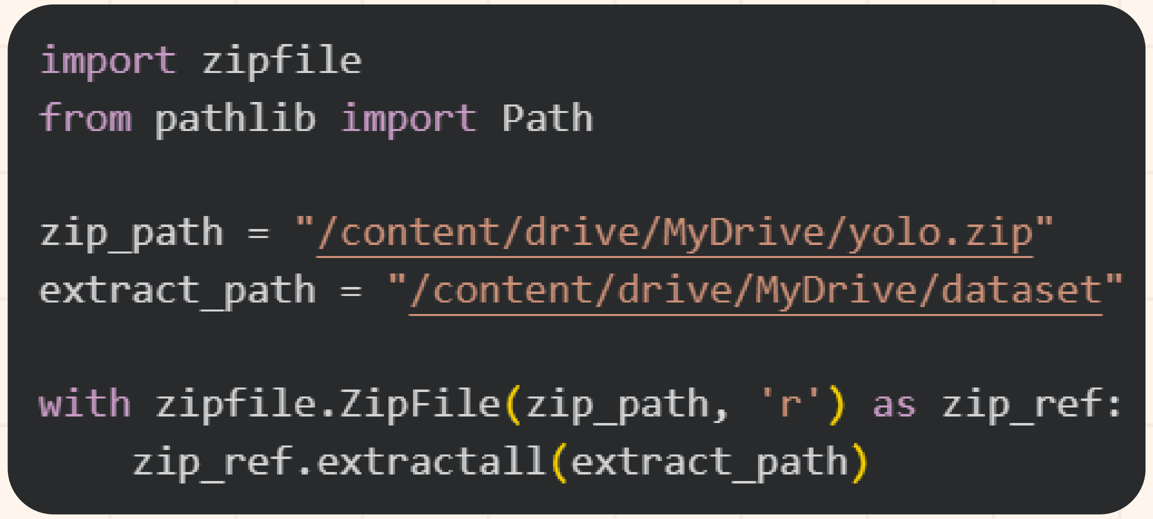
- Folder name = class label
- Images only (no annotation files)

Dataset Setup

Dataset Structure Requirement (YOLO Classification)



```
dataset_yolo/                ├──── cat/
   ├──── train/              ├──── cow/
   ├──── val/       →        ├──── dog/
   └──── test/               ├──── lamb/
                             └──── zebra/
```



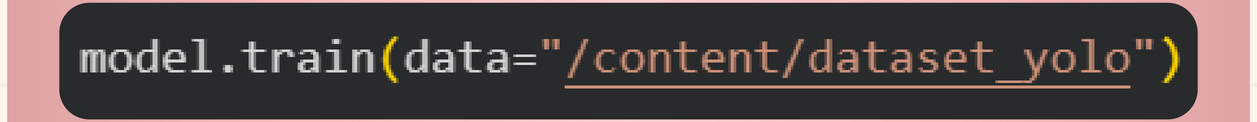
```python
import zipfile
from pathlib import Path

zip_path = "/content/drive/MyDrive/yolo.zip"
extract_path = "/content/drive/MyDrive/dataset"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)
```

→ Unzips dataset containing class folders only.

Auto Dataset Splitting

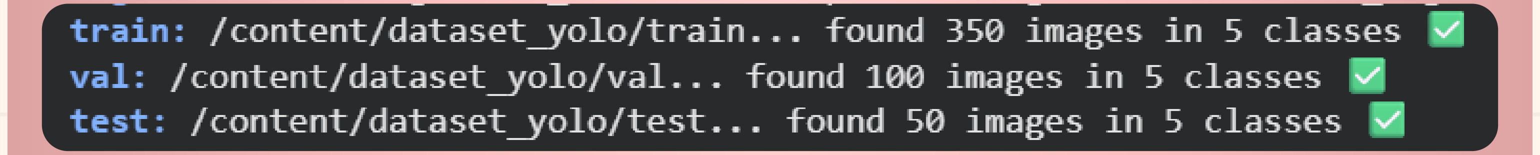


```python
model.train(data="/content/dataset_yolo")
```

✓ No manual splitting required

→
- **70% Training**
- **20% Validation**
- **10% Testing**

Output



```
train: /content/dataset_yolo/train... found 350 images in 5 classes ✅
val: /content/dataset_yolo/val... found 100 images in 5 classes ✅
test: /content/dataset_yolo/test... found 50 images in 5 classes ✅
```

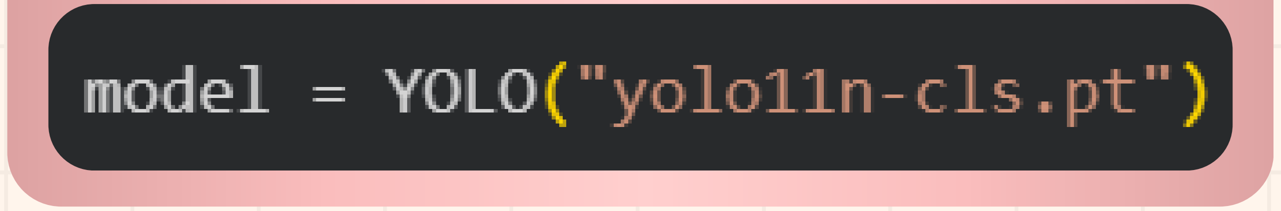
train: 350 images in 5 classes
val: 100 images in 5 classes
test: 50 images in 5 classes

✓ Balanced dataset
✓ Each class contains equal samples

Mubashshira Kaisar 2221070642 | **Noshin Nawar 2221507042** | **Group No. - 1** | **Md Shihabul Alam Shihab 2233115642** | **Md. Thaminuzzaman 2312877642**

YOLOV11 MODEL

Model Used



```
model = YOLO("yolo11n-cls.pt")
```

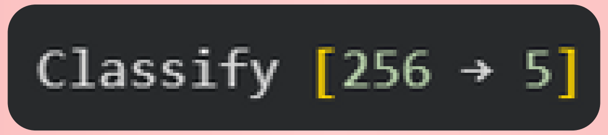
→

✓ YOLOv11 Nano – Classification variant
✓ Lightweight & efficient for CPU
✓ Pretrained on ImageNet

Architecture Overview

YOLOv11-cls is a Convolutional Neural Network (CNN) consisting of:

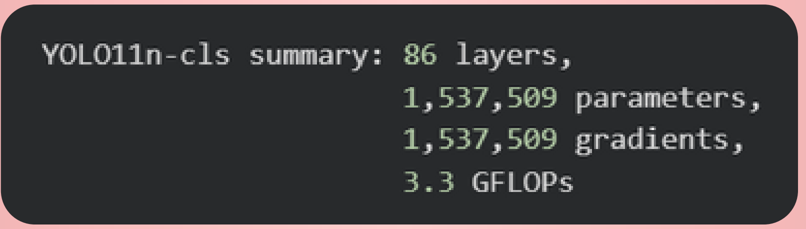
1. Convolution Layers
   - Extract low-level features (edges, textures)
2. C3k2 & C2PSA Blocks
   - Learn complex spatial patterns
3. Classification Head



```
Classify [256 → 5]
```

Outputs probability for 5 animal classes

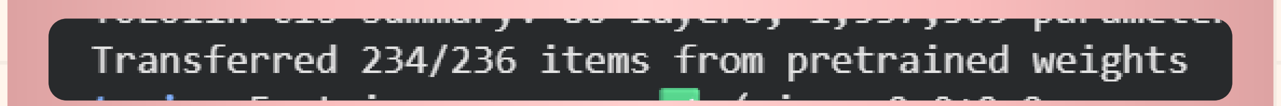
Model Summary Output



```
YOLO11n-cls summary: 86 layers,
                     1,537,509 parameters,
                     1,537,509 gradients,
                     3.3 GFLOPs
```

- Small model
- Fast inference
- Suitable for real-time use

Transfer Learning

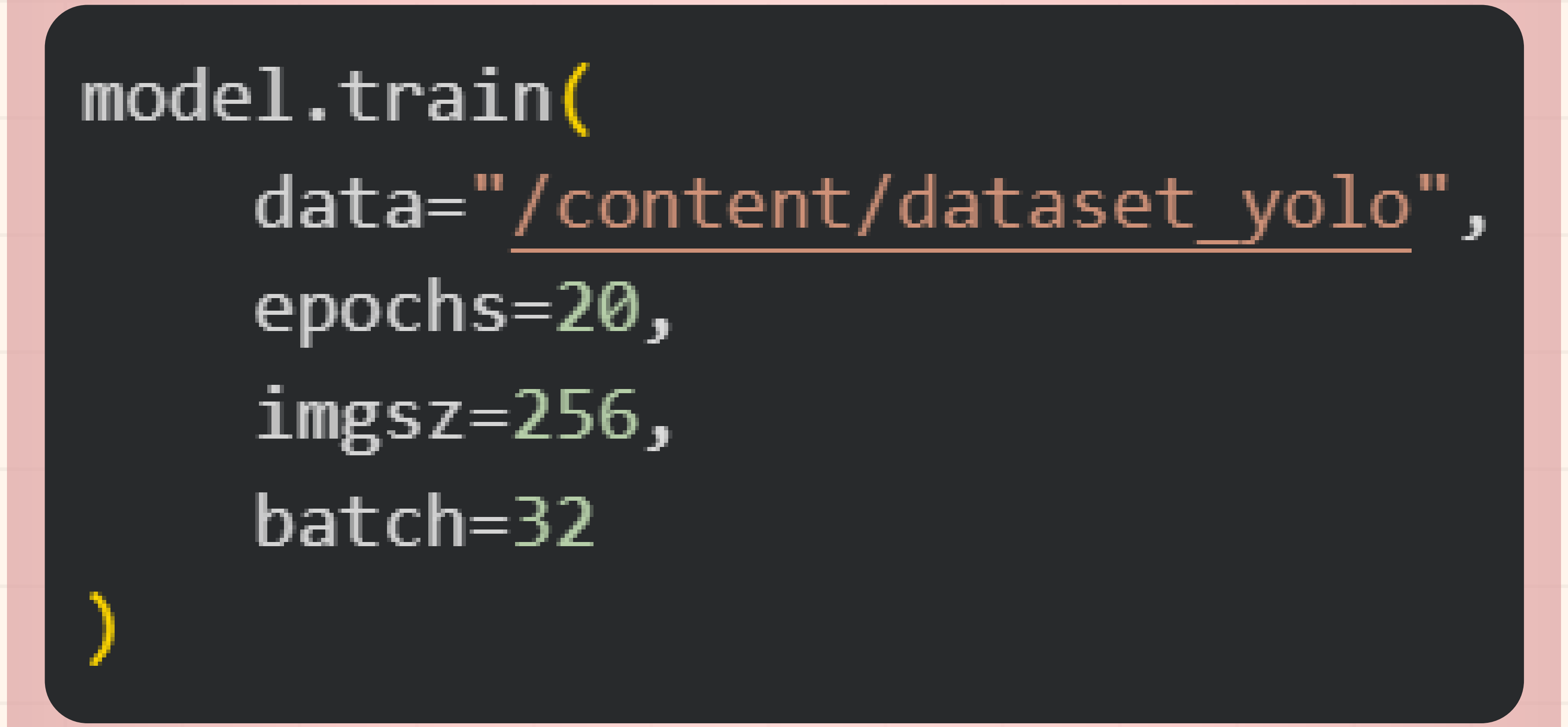


```
Transferred 234/236 items from pretrained weights
```

- Uses learned features from large datasets
- Faster convergence
- Higher accuracy with fewer epochs

TRAINING & PERFORMANCE ANALYSIS

Training Configuration



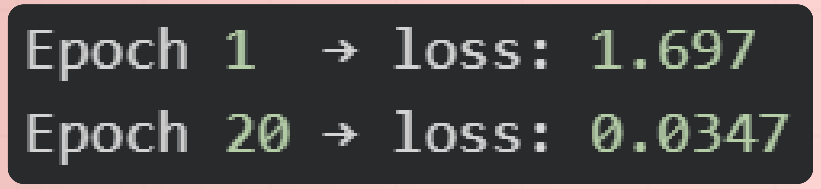
```
model.train(
    data="/content/dataset_yolo",
    epochs=20,
    imgsz=256,
    batch=32
)
```

Parameter	Purpose
epochs=20	Number of full training cycles
imgsz=256	Image resized to 256×256
batch=32	Images per iteration
optimizer	AdamW (auto-selected)

TRAINING & PERFORMANCE ANALYSIS (CONT.)

Training Progress

Loss Reduction

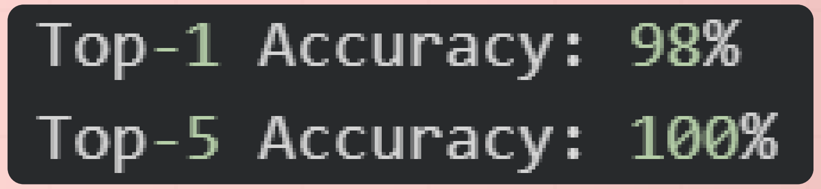


```
Epoch 1  → loss: 1.697
Epoch 20 → loss: 0.0347
```

✓ Indicates successful learning

✓ No overfitting observed

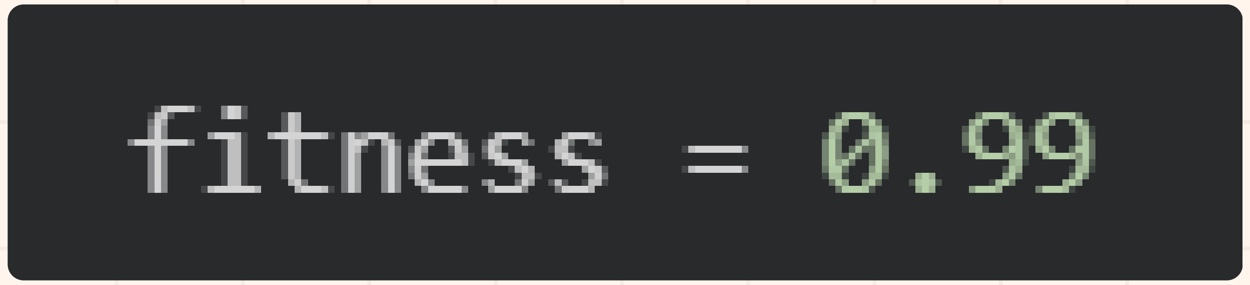
Accuracy Metrics



```
Top-1 Accuracy: 98%
Top-5 Accuracy: 100%
```

- Top-1 → Correct class ranked first
- Top-5 → Correct class within top 5 predictions
- Excellent classification performance

Best Model Shows →



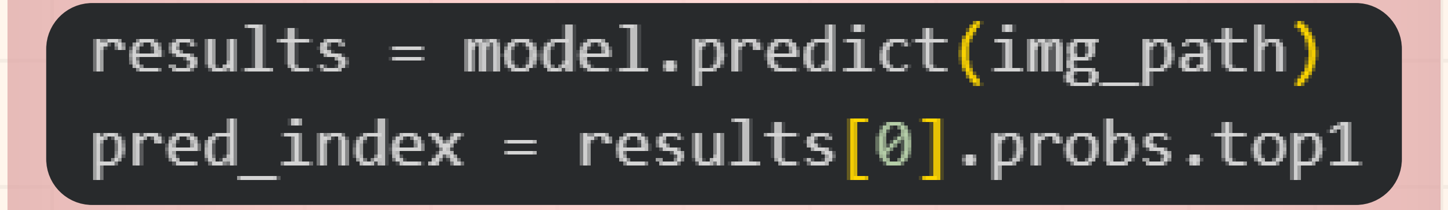
```
fitness = 0.99
```

→

- Combined metric for performance
- Best weights saved automatically

EVALUATION, TESTING & PREDICTION

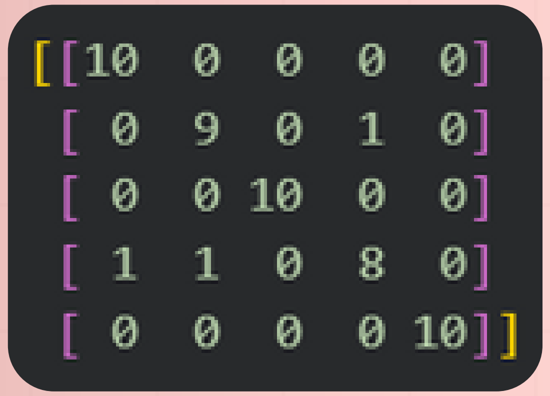
Evaluation on Test Dataset



```
results = model.predict(img_path)
pred_index = results[0].probs.top1
```

✓ Predicts class with highest probability
✓ Compared against true labels

Confusion Matrix



```
[[10  0  0  0  0]
 [ 0  9  0  1  0]
 [ 0  0 10  0  0]
 [ 1  1  0  8  0]
 [ 0  0  0  0 10]]
```

Interpretation:

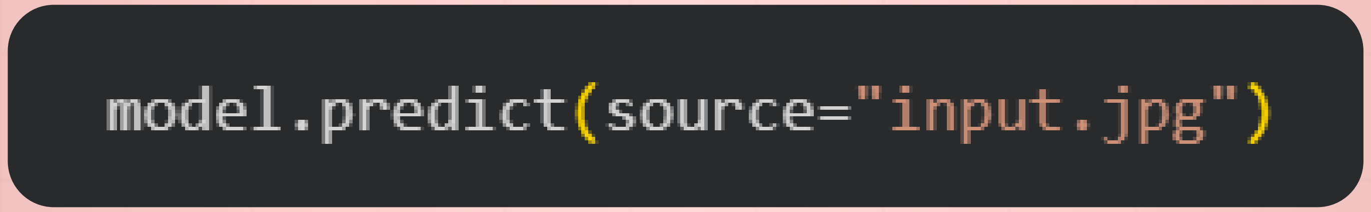
- **Diagonal values = Correct predictions**
- **Minor confusion between cow ↔ lamb**
- **Dog & zebra achieved 100% accuracy**

Classification Report Highlights

Overall Accuracy → 94%

✓ Strong generalization
✓ Slight ambiguity in visually similar animals

Single Image Prediction



```
model.predict(source="input.jpg")
```

✓ Model correctly classified unseen image
✓ Visual confidence scores displayed

Mubashshira Kaisar
2221070642

Noshin Nawar
2221507042

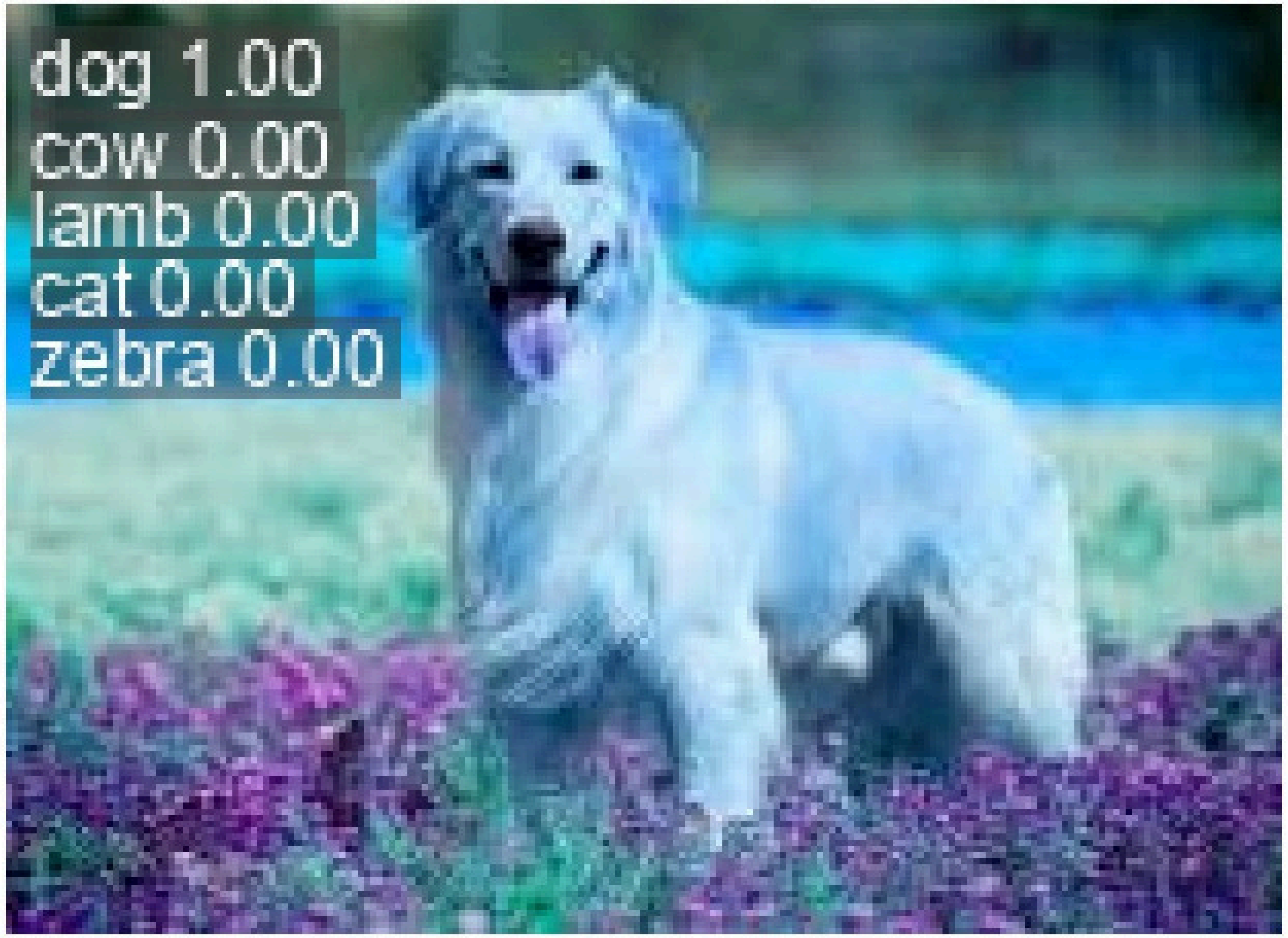
Group No. - 1

Md Shihabul Alam Shihab
2233115642

Md. Thaminuzzaman
2312877642

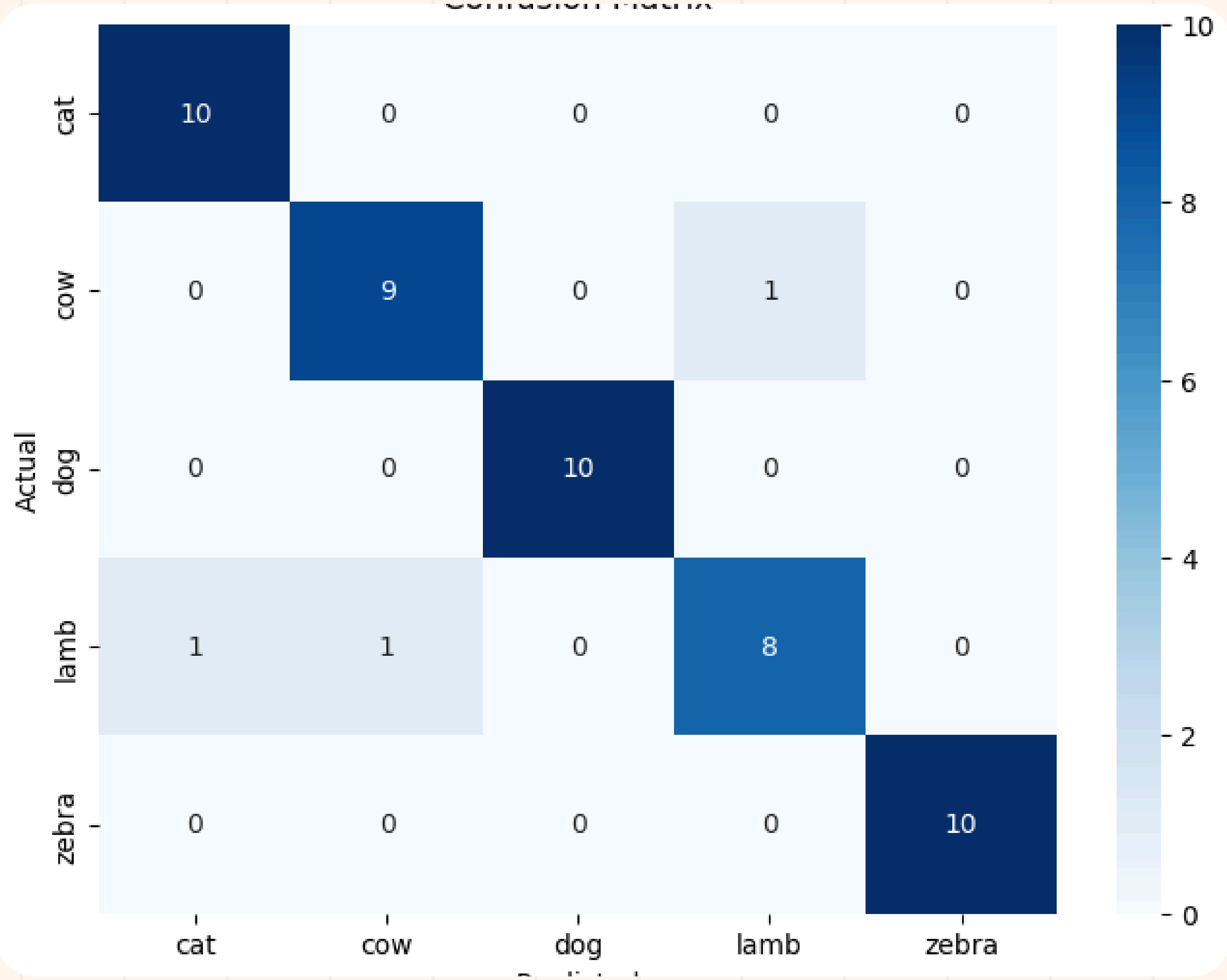
OUTPUT SCREENSHOTS

Output



```
dog 1.00
cow 0.00
lamb 0.00
cat 0.00
zebra 0.00
```

Confusion Matrix



Actual	cat	cow	dog	lamb	zebra
cat	10	0	0	0	0
cow	0	9	0	1	0
dog	0	0	10	0	0
lamb	1	1	0	8	0
zebra	0	0	0	0	10

THANK YOU

Mubashshira Kaisar
2221070642

Noshin Nawar
2221507042

Group No. - 1

Md Shihabul Alam Shihab
2233115642

Md. Thaminuzzaman
2312877642