# INVESTIGATING THE RELATIONSHIP BETWEEN AI MODEL COMPLEXITY AND ACCURACY FOR IMAGE CLASSIFICATION

PRESENTER: NIKITA FERENTS

21 June, 2024

# INTRODUCTION

Project Objective: To analyze the impact of model complexity, type, source, training time, and size on the accuracy of image classification using the CIFAR-10 dataset.

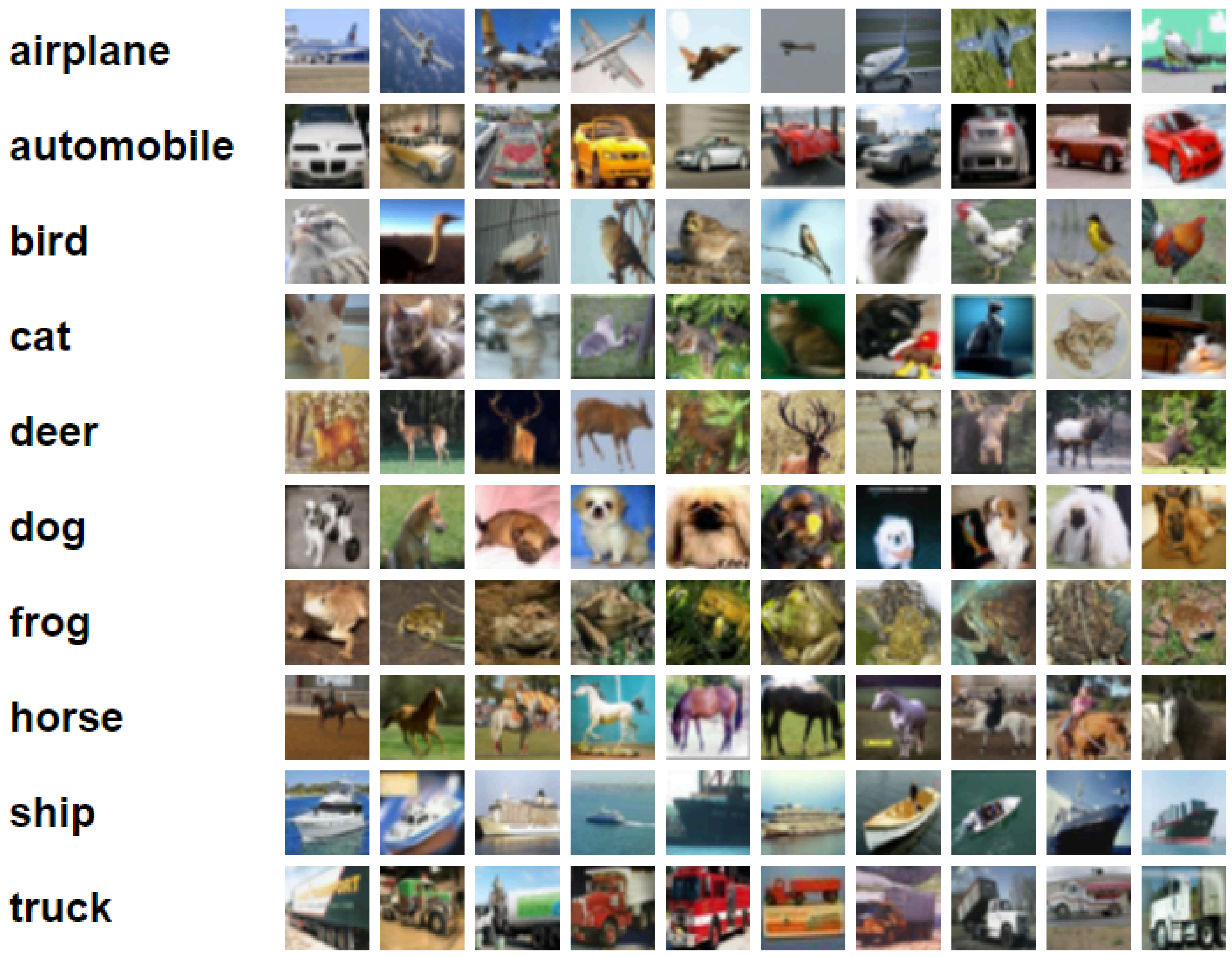
# DATASET OVERVIEW

## CIFAR-10 DATASET

LOW-RESOLUTION (32X32 PIXELS) IMAGES CATEGORIZED INTO 10 CLASSES.

CATEGORIES: AIRPLANE, CAR, BIRD, CAT, DEER, DOG, FROG, HORSE, SHIP, TRUCK

SOURCE: PAPERS WITH CODE - CIFAR-10



CIFAR-10 (Canadian Institute for Advanced Research, 10 classes)

## Benchmarks

Edit

| Trend | Task | Dataset Variant | Best Model | Paper | Code |
|---|---|---|---|---|---|
| | **Image Classification** | CIFAR-10 | efficient adaptive ensembling | | |
| | **Image Generation** | CIFAR-10 | StyleSAN-XL | | |
| | **Long-tail Learning** | CIFAR-10-LT (ρ=10) | GLMC+MaxNorm | | |
| | **Semi-Supervised Image Classification** | CIFAR-10, 4000 Labels | SemCo | | |
| | **Neural Architecture Search** | CIFAR-10 | NAT-M4 | | |

Show all 104 benchmarks

## Papers

Search for a paper or author

| Paper | Code | Results | Date | Stars |
|---|---|---|---|---|
| Learning Transferable Architectures for Scalable Image Recognition | | | 21 лип. 2017 | 76 744 |
| Deep Residual Learning for Image Recognition | | | 10 груд. 2015 | 76 743 |
| Meta-Learning Update Rules for Unsupervised Representation Learning | | – | 31 бер. 2018 | 76 740 |
| AutoAugment: Learning Augmentation Policies from Data | | | 24 трав. 2018 | 76 740 |
| Density estimation using Real NVP | | | 27 трав. 2016 | 76 740 |
| Progressive Neural Architecture Search | | | 2 груд. 2017 | 76 740 |

# SOURCE OVERVIEW

Papers with Code connects research papers with their implementations, datasets, and benchmarks. It allows users to access cutting-edge AI methods, compare model performances, and explore datasets like CIFAR-10. This platform is essential for researchers and developers aiming to implement state-of-the-art techniques efficiently.

# MODELS USED

## 01
### SIMPLE CNN

Architecture: Basic convolutional layers, pooling, dropout, and dense layers.
Purpose: Baseline model for comparison.

## 02
### VGG-16 INSPIRED MODEL

Architecture: Deeper network with multiple convolutional layers, designed for high-resolution images.
Purpose: Evaluate performance on low-resolution images.

## 03
### YOLOv8 MODEL

Architecture: Efficient model known for real-time object detection, repurposed for classification.
Purpose: Test versatility and performance.

# ▪ SIMPLE CNN ARCHITECTURE

```
modeleasy = keras.models.Sequential()
modeleasy.add(keras.layers.Conv2D(32, (3, 3), activation='relu',
padding='same', input_shape=(32,32,3)))
modeleasy.add(keras.layers.Conv2D(32, (3, 3), activation='relu',
padding='same'))
modeleasy.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
modeleasy.add(keras.layers.Dropout(0.25))
modeleasy.add(keras.layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
modeleasy.add(keras.layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
modeleasy.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
modeleasy.add(keras.layers.Dropout(0.25))
modeleasy.add(keras.layers.Flatten())
modeleasy.add(keras.layers.Dense(512, activation='relu'))
modeleasy.add(keras.layers.Dropout(0.5))
modeleasy.add(keras.layers.Dense(10, activation='softmax'))
modeleasy.summary()
```

**Training**: 20 epochs, batch size of 32, validation split of 20%.

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 32, 32, 32) | 896 |
| conv2d_1 (Conv2D) | (None, 32, 32, 32) | 9248 |
| max_pooling2d (MaxPooling2D) | (None, 16, 16, 32) | 0 |
| dropout (Dropout) | (None, 16, 16, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 16, 16, 64) | 18496 |
| conv2d_3 (Conv2D) | (None, 16, 16, 64) | 36928 |
| max_pooling2d_1 (MaxPooling2D) | (None, 8, 8, 64) | 0 |
| dropout_1 (Dropout) | (None, 8, 8, 64) | 0 |
| flatten (Flatten) | (None, 4096) | 0 |
| dense (Dense) | (None, 512) | 2097664 |
| dropout_2 (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 10) | 5130 |

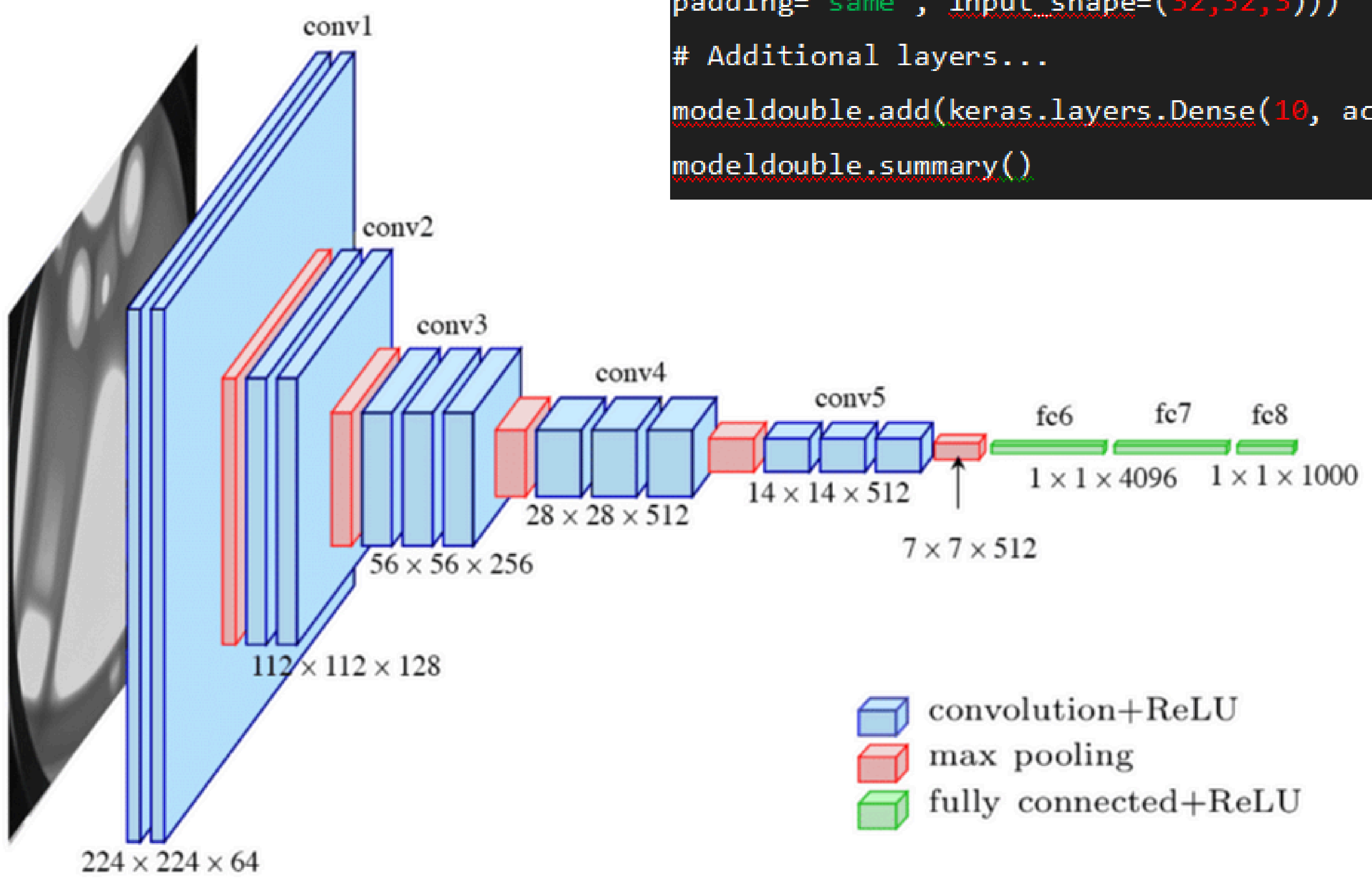
Total params: 2168362 (8.27 MB)
Trainable params: 2168362 (8.27 MB)
Non-trainable params: 0 (0.00 Byte)

**Evaluation:**

- Loss: 0.7359
- Accuracy: 0.7728

# ▪ VGG-16 INSPIRED MODEL

```
modeldouble = keras.models.Sequential()
modeldouble.add(keras.layers.Conv2D(64, (3, 3), activation='relu',
padding='same', input_shape=(32,32,3)))
# Additional layers...
modeldouble.add(keras.layers.Dense(10, activation='softmax'))
modeldouble.summary()
```




**Training**:
20 epochs, batch size of 32, validation split of 20%.

**Evaluation:**
Loss: 2.3026
Accuracy: 0.1000

**Conclusion**:
Designed for high-resolution images, struggled with low-resolution CIFAR-10.

# YOLO MODEL OVERVIEW

YOLO (You Only Look Once) is a state-of-the-art model known for real-time object detection and classification. It is designed for efficiency and speed, making it suitable for a wide range of applications. YOLO models, including the latest YOLOv8, offer high accuracy and versatility, performing well on both low-resolution datasets like CIFAR-10 and more complex, high-resolution images. YOLO states this makes it is excellent choice for various image classification tasks.

# ▪ YOLO MODEL

YOLO (You Only Look Once) is highly effective for image classification, offering models of varying sizes and complexities. These models are pretrained on the ImageNet dataset, ensuring high performance across different image classification tasks.

| Model | size (pixels) | acc top1 | acc top5 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) at 640 |
|---|---|---|---|---|---|---|---|
| YOLOv8n-cls | 224 | 69.0 | 88.3 | 12.9 | 0.31 | 2.7 | 4.3 |
| YOLOv8s-cls | 224 | 73.8 | 91.7 | 23.4 | 0.35 | 6.4 | 13.5 |
| YOLOv8m-cls | 224 | 76.8 | 93.5 | 85.4 | 0.62 | 17.0 | 42.7 |
| YOLOv8l-cls | 224 | 76.8 | 93.5 | 163.0 | 0.87 | 37.5 | 99.7 |
| YOLOv8x-cls | 224 | 79.0 | 94.6 | 232.0 | 1.01 | 57.4 | 154.8 |

YOLOv8n-cls is the smallest and fastest model in the YOLOv8 classification series. It is particularly suitable for the CIFAR-10 dataset due to its:

- Efficiency
- Adequate Accuracy
- Resource-Friendly

This makes YOLOv8n-cls a practical choice for projects involving low-resolution image classification, where both performance and efficiency are crucial.

- **YOLOv8n-cls**: Smallest and fastest, with lower accuracy.
- **YOLOv8s-cls**: Slightly larger with improved accuracy.
- **YOLOv8m-cls**: Medium-sized, balancing speed and accuracy.
- **YOLOv8l-cls**: Large model with high accuracy but slower.
- **YOLOv8x-cls**: Largest and most accurate, but slowest.

# ▪ YOLO MODEL

**Training**:

5 epochs, image size of 32, validation split of 20%.

```
!pip install ultralytics
```

```python
from ultralytics import YOLO
model = YOLO("yolov8n-cls.pt")
results = model.train(data="cifar10", epochs=5, imgsz=32)
```

```
speed: {'preprocess': 0.0007508039474487304, 'inference': 0.6635741472244262, 'loss': 0.00010340213775634765, 'postprocess': 8.27789306640625e-05}
task: 'classify'
top1: 0.736299991607666
top5: 0.9825000166893005
```

**Evaluation**:

Top-1 Accuracy: 0.7363

Top-5 Accuracy: 0.9825

**Advantages**:

High versatility and robust performance, even on low-resolution images.

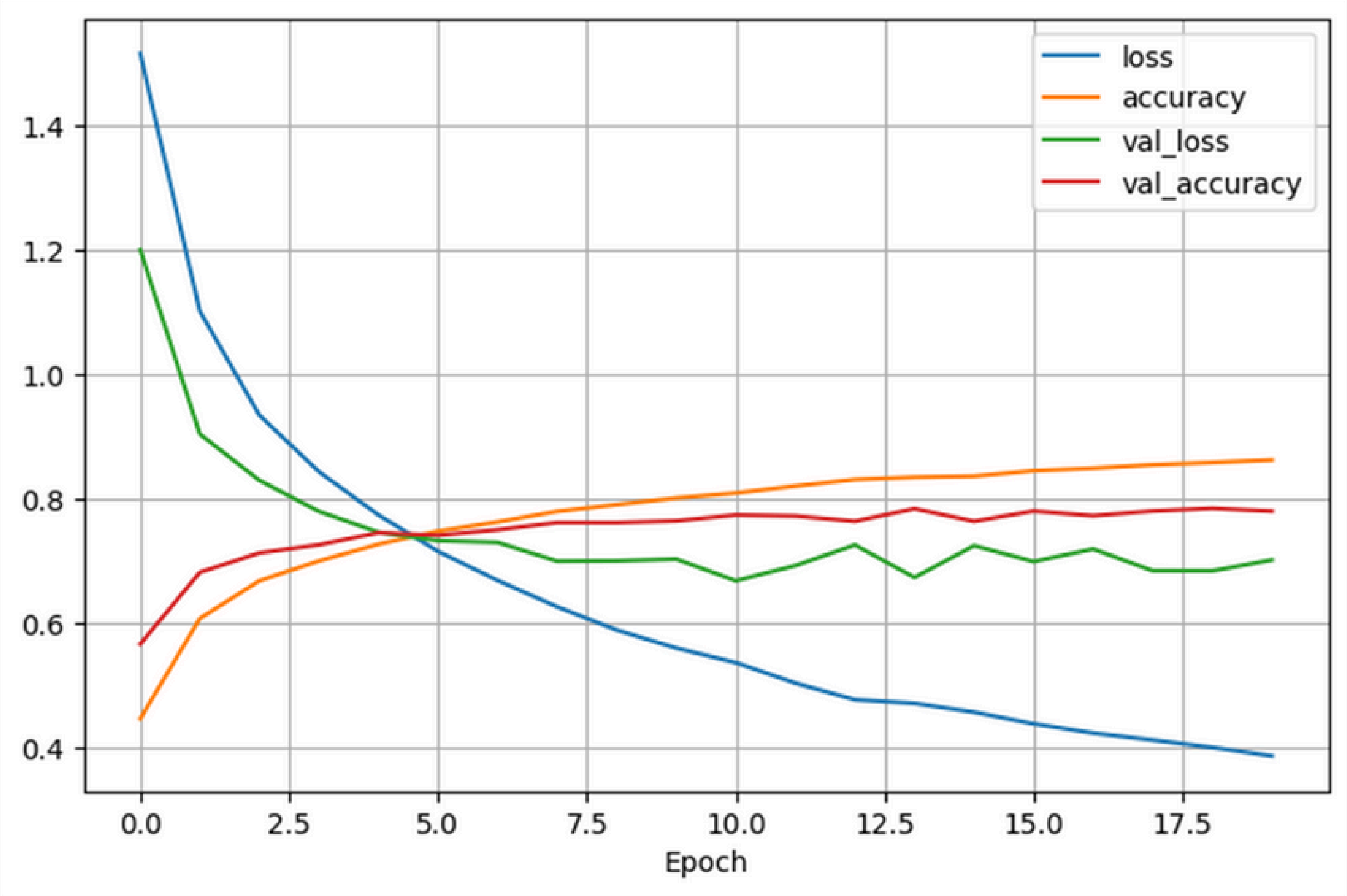
# VISUALIZING TRAINING HISTORY

Training History Visualization:

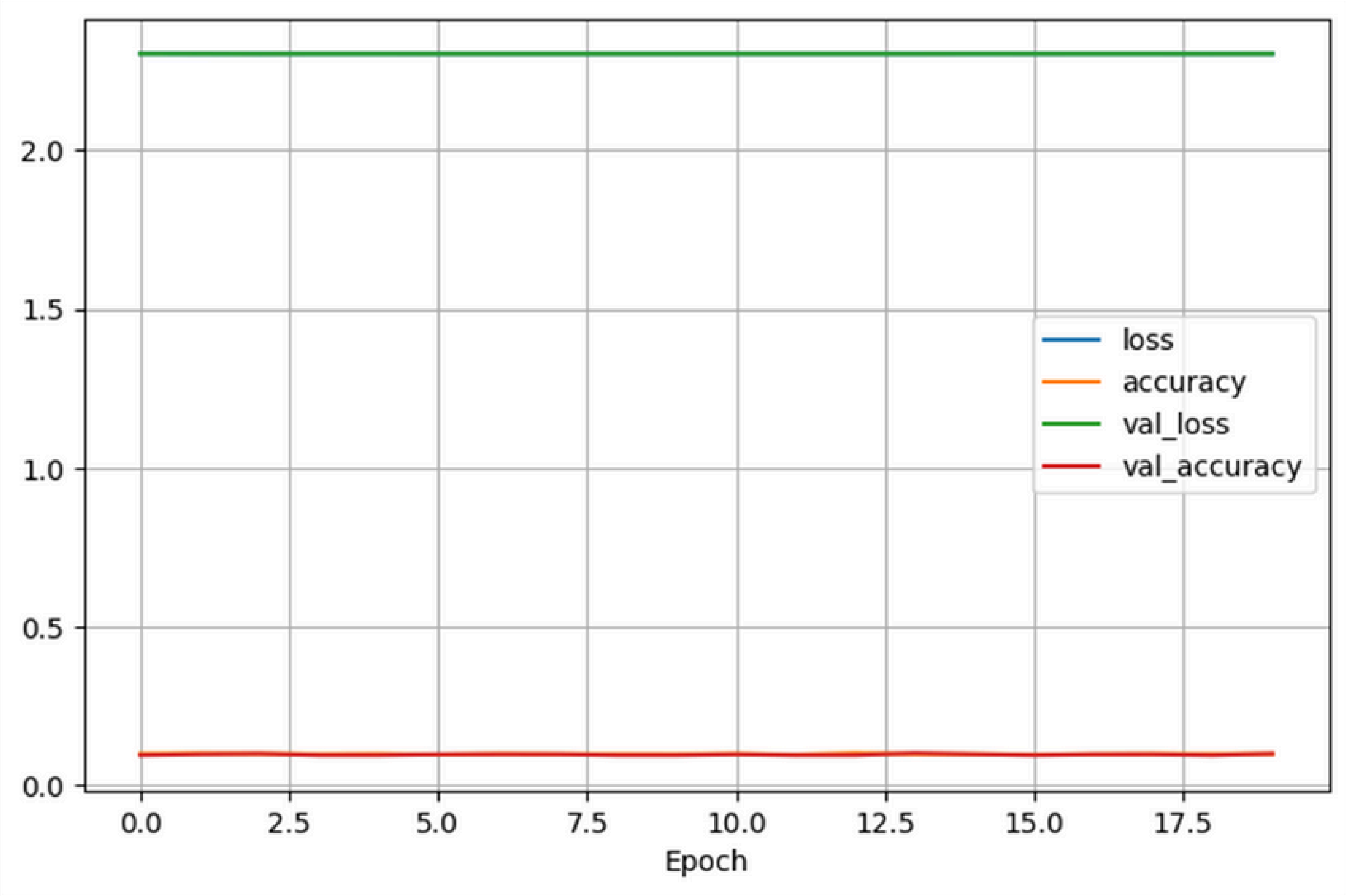
**Simple CNN and VGG-16**: Plotted accuracy and loss over epochs.

**YOLO Model**: Detailed evaluation metrics.

## SIMPLE CNN




## VGG-16 INSPIRED




## YOLO MODEL

```
speed: {'preprocess': 0.0007508039474487304, 'inference': 0.6635741472244262, 'loss': 0.00010340213775634765, 'postprocess': 8.27789306640625e-05}
task: 'classify'
top1: 0.736299991607666
top5: 0.9825000166893005
```

# CONCLUSIONS

Key Findings:

- Simple CNN: Balanced performance, suitable for low-resolution images.
- VGG-16 Inspired Model: Underperformed due to the low-resolution nature of CIFAR-10.
- YOLO Model: Demonstrated strong performance and versatility, especially in Top-5 accuracy.

Overall, this project highlights the importance of matching model complexity and architecture to the characteristics of the dataset. While deep models like VGG-16 are powerful for high-resolution images, simpler models or versatile models like YOLO can be more effective for lower-resolution tasks such as those presented by CIFAR-10.

# THANK YOU

21June, 2024

# INVESTIGATING THE RELATIONSHIP BETWEEN AI MODEL COMPLEXITY AND ACCURACY FOR IMAGE CLASSIFICATION

PRESENTER: NIKITA FERENTS

21 June, 2024