AMERICAN UNIVERSITY OF PHNOM PENH

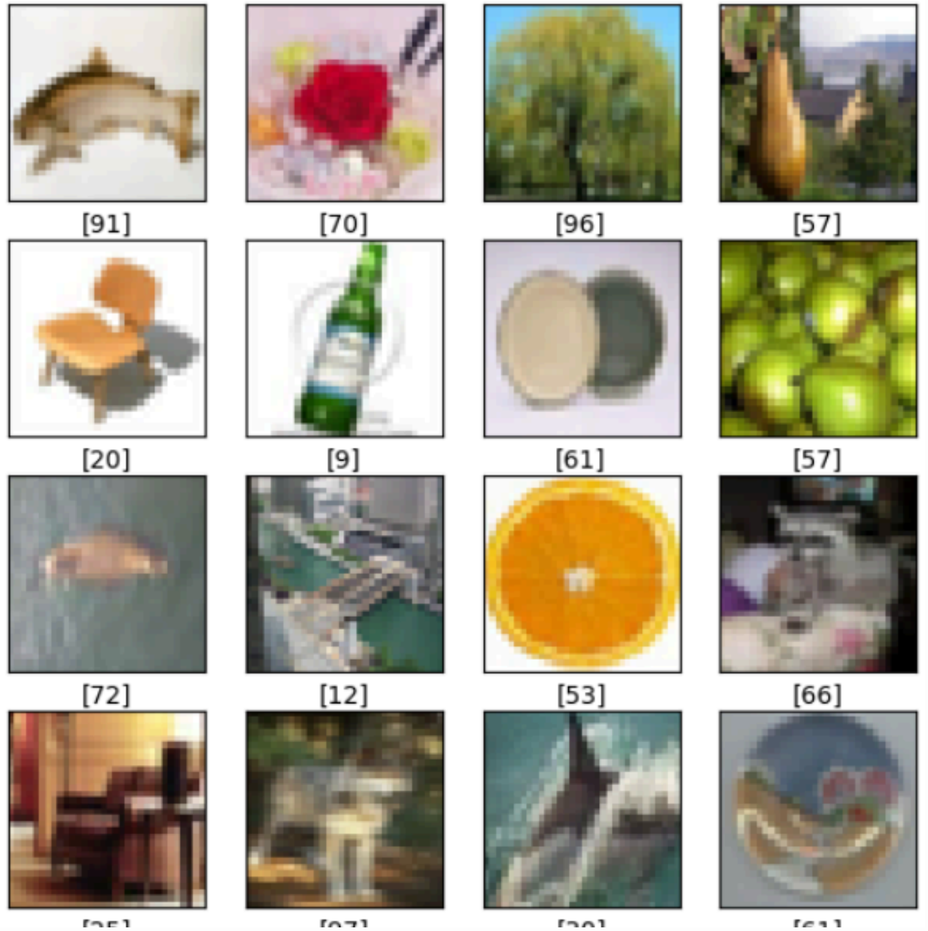
STUDY LOCALLY. LIVE GLOBALLY.

Final Project

Image Classification

Group 11:

1. Oeng Meily
2. Uchita Hikaru
3. Somoeurn Virakden



Overview

- Introduction
- Dataset and Results
- Fully connected neural network architecture
- Convolutional Neural Network
- Demonstration

Indroduction

- Image Classification
- Project Objective

Image Classification

Computer Vision

- Object Detection
- **Image Classification**
- Image reconstruction
- Face Recognition
- Semantic Segmentation

- We will tackle one of the computer vision task is Image Classification.
- The importance of image classification in today's from catalog cifa100.

Project Objective

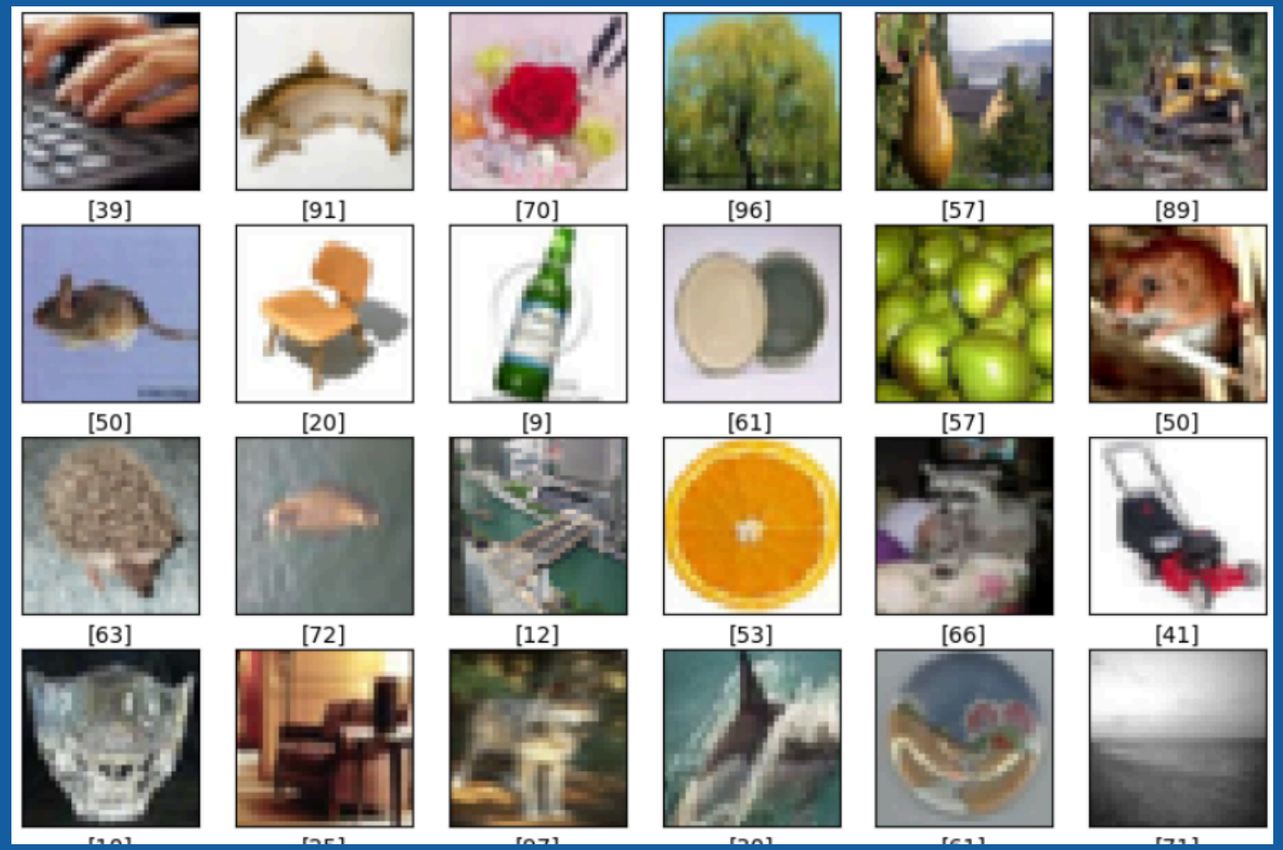
- Develop a CNN-based model for accurate image classification with dataset cifar100.
- Utilize TensorFlow and Keras for model training and optimization.
- Deploy the model using Gradio for real-time image classification and feedback.

Dataset and Results

- Preprocessing
- Data Limitation

Dataset

some of these images and their corresponding training labels look like.



Dataset

CIFAR-10 has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Fully connected neural network

- Model Design
- Training Process
- Model Evaluation

Model Design

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 3072)              0

 dense (Dense)               (None, 1024)              3146752

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 100)               25700

=================================================================
Total params: 3828580 (14.60 MB)
Trainable params: 3828580 (14.60 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Training Process

```python
early_stopping = EarlyStopping(monitor='loss', patience=5, restore_best_weights=True)

BATCH_SIZE = 128
EPOCHS = 100

model.fit(train_images, train_labels, batch_size=BATCH_SIZE, epochs=EPOCHS, callbacks=[early_stopping])
```

```
Epoch 73/100
391/391 [==============================] - 2s 4ms/step - loss: 0.6472 - accuracy: 0.8079
Epoch 74/100
391/391 [==============================] - 2s 4ms/step - loss: 0.6489 - accuracy: 0.8064
Epoch 75/100
391/391 [==============================] - 2s 4ms/step - loss: 0.6421 - accuracy: 0.8079
Epoch 76/100
391/391 [==============================] - 2s 4ms/step - loss: 0.6393 - accuracy: 0.8094
Epoch 77/100
391/391 [==============================] - 2s 6ms/step - loss: 0.6384 - accuracy: 0.8099

```

Evaluation

```
1  test_loss, test_acc = model.evaluate(test_images, test_labels) # TODO
2  print('Test loss:', test_loss, 'Test accuracy:', test_acc)
3
```

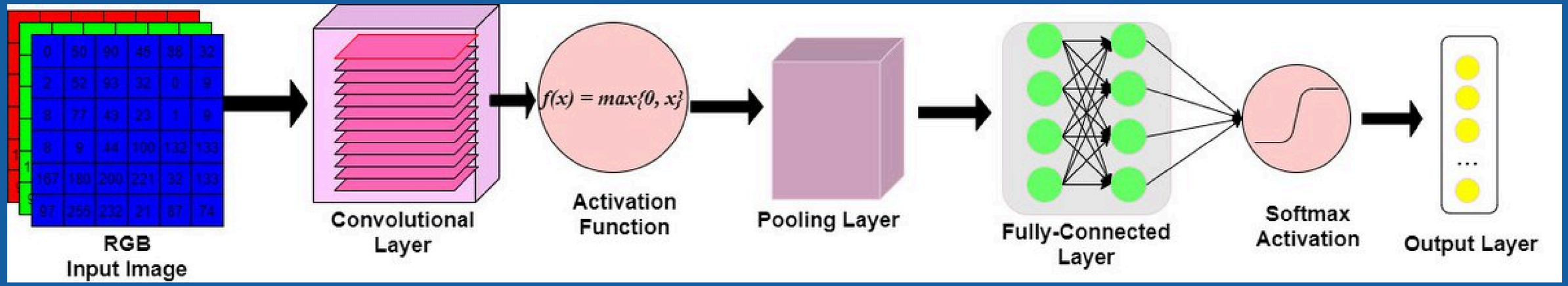
```
1  313/313 [==============================] - 1s 3ms/step - loss: 5.7446 - accuracy: 0.2702
2  Test loss: 5.744603157043457 Test accuracy: 0.2702000141143799
3
```

Convolutional Neural Network

- Methodology
- Model Design
- Training Process
- Model Evaluation

Methodology

Model Design



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 128)       3584

 conv2d_7 (Conv2D)           (None, 30, 30, 128)       147584

 max_pooling2d_3 (MaxPoolin  (None, 15, 15, 128)       0
 g2D)

 dropout_4 (Dropout)         (None, 15, 15, 128)       0

 conv2d_8 (Conv2D)           (None, 15, 15, 256)       295168

 conv2d_9 (Conv2D)           (None, 13, 13, 256)       590080

 max_pooling2d_4 (MaxPoolin  (None, 6, 6, 256)         0
 g2D)

 dropout_5 (Dropout)         (None, 6, 6, 256)         0

 conv2d_10 (Conv2D)          (None, 6, 6, 512)         1180160

 conv2d_11 (Conv2D)          (None, 4, 4, 512)         2359808

 max_pooling2d_5 (MaxPoolin  (None, 2, 2, 512)         0
 g2D)

 dropout_6 (Dropout)         (None, 2, 2, 512)         0

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 1024)              2098176

 dropout_7 (Dropout)         (None, 1024)              0

 dense_3 (Dense)             (None, 256)               262400

 dropout_8 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 100)               25700

=================================================================
Total params: 6962660 (26.56 MB)
Trainable params: 6962660 (26.56 MB)
Non-trainable params: 0 (0.00 Byte)
```

Model Design

Training Process & Evaluation

```
BATCH_SIZE = 128
early_stopping = EarlyStopping(monitor='loss', patience=3, restore_best_weights=True)
EPOCHS = 100

history = cnn_model.fit(
    train_images,
    train_labels,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=[early_stopping]
)
```

Training Process & Evaluation

```
Epoch 1/10
391/391 [==============================] - 24s 62ms/step - loss: 1.4566 - accuracy: 0.5872
Epoch 2/10
391/391 [==============================] - 24s 62ms/step - loss: 1.3644 - accuracy: 0.6082
Epoch 3/10
391/391 [==============================] - 24s 62ms/step - loss: 1.2901 - accuracy: 0.6261
Epoch 4/10
391/391 [==============================] - 24s 62ms/step - loss: 1.2232 - accuracy: 0.6426
Epoch 5/10
391/391 [==============================] - 24s 62ms/step - loss: 1.1434 - accuracy: 0.6643
Epoch 6/10
391/391 [==============================] - 24s 62ms/step - loss: 1.0850 - accuracy: 0.6769
Epoch 7/10
391/391 [==============================] - 24s 62ms/step - loss: 1.0295 - accuracy: 0.6915
Epoch 8/10
391/391 [==============================] - 24s 62ms/step - loss: 0.9627 - accuracy: 0.7078
Epoch 9/10
391/391 [==============================] - 24s 62ms/step - loss: 0.9166 - accuracy: 0.7225
Epoch 10/10
391/391 [==============================] - 24s 62ms/step - loss: 0.8797 - accuracy: 0.7314

```

Evaluation

```
313/313 [==============================] - 3s 8ms/step - loss: 2.2440 - accuracy: 0.4867
Test loss: 2.2440364360809326 Test accuracy: 0.48669999837875366

```

Demonstration

Thank you!