COURSE: DSA4050

COURSE TITLE: Deep Learning for Computer Vision

INSTRUCTOR: DR. EDWARD OMBUI

Week 4 Practical Exercises for Inception, ResNet, DenseNet Architectures

---

Step-by-step practical exercises for implementing Inception, ResNet, and DenseNet architectures using TensorFlow and Keras. Each exercise includes loading a pre-trained model, modifying it for a new task, training, and evaluating performance.

Architecture	Dataset	Task
InceptionV3	Cats vs. Dogs	Binary Image Classification
ResNet50	CIFAR-10	Multi-Class Image Classification
DenseNet121	Chest X-ray (Pneumonia)	Medical Image Classification

Prerequisites

Before starting, ensure you have TensorFlow installed. If using Google Colab, run the following:

!pip install tensorflow

Exercise 1: Implementing InceptionV3 for Image Classification

Task: Fine-tune a pre-trained InceptionV3 model on the Cats vs. Dogs dataset.

Step 1: Import Necessary Libraries

```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Step 2: Load Pre-trained InceptionV3 Model

```
# Load pre-trained InceptionV3 model without the top classification layer
base_model = InceptionV3(weights='imagenet', include_top=False,
input_shape=(150, 150, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False
```

Step 3: Add Custom Classification Layers

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dense(1, activation='sigmoid')  # Binary classification (Cats vs. Dogs)

# Define new model
model = Model(inputs=base_model.input, outputs=x)
```

Step 4: Compile the Model

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

Step 5: Prepare Dataset and Data Augmentation

```
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    'dataset/cats_vs_dogs/', target_size=(150, 150), batch_size=32,
class_mode='binary', subset='training')

val_generator = train_datagen.flow_from_directory(
    'dataset/cats_vs_dogs/', target_size=(150, 150), batch_size=32,
class_mode='binary', subset='validation')
```

Step 6: Train the Model

```
history = model.fit(train_generator, epochs=10,
validation_data=val_generator)
```

Step 7: Evaluate and Save the Model

```
model.evaluate(val_generator)
model.save("inception_cats_vs_dogs.h5")
```

Exercise 2: Implementing ResNet50 for Multi-Class Classification

Task: Fine-tune a ResNet50 model on the CIFAR-10 dataset (10 classes).

Step 1: Import Necessary Libraries

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
```

Step 2: Load and Preprocess CIFAR-10 Dataset

```
# Load CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) =
cifar10.load_data()

# Normalize pixel values
train_images, test_images = train_images / 255.0, test_images / 255.0

# Convert labels to one-hot encoding
train_labels, test_labels = to_categorical(train_labels, 10),
to_categorical(test_labels, 10)
```

Step 3: Load Pre-trained ResNet50 Model

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(32,
32, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False
```

Step 4: Add Custom Classification Layers

```
x = base_model.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
x = Dense(10, activation='softmax')  # 10 classes

model = Model(inputs=base_model.input, outputs=x)
```

Step 5: Compile the Model

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

Step 6: Train the Model

```
history = model.fit(train_images, train_labels, epochs=10,
validation_data=(test_images, test_labels))
```

Step 7: Evaluate and Save the Model

```
model.evaluate(test_images, test_labels)
model.save("resnet_cifar10.h5")
```

Exercise 3: Implementing DenseNet121 for Medical Image Classification

Task: Fine-tune a DenseNet121 model on a custom Chest X-ray dataset for pneumonia detection.

Step 1: Import Necessary Libraries

```
import tensorflow as tf
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Step 2: Load Pre-trained DenseNet121 Model

```
base_model = DenseNet121(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False
```

Step 3: Add Custom Classification Layers

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dense(1, activation='sigmoid')  # Binary classification

model = Model(inputs=base_model.input, outputs=x)
```

Step 4: Compile the Model

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

Step 5: Prepare Medical Dataset

```
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
```

```
train_generator = train_datagen.flow_from_directory(
    'dataset/chest_xray/', target_size=(224, 224), batch_size=32,
class_mode='binary', subset='training')

val_generator = train_datagen.flow_from_directory(
    'dataset/chest_xray/', target_size=(224, 224), batch_size=32,
class_mode='binary', subset='validation')
```

Step 6: Train the Model

```
history = model.fit(train_generator, epochs=10,
validation_data=val_generator)
```

Step 7: Evaluate and Save the Model

```
model.evaluate(val_generator)
model.save("densenet_pneumonia.h5")
```