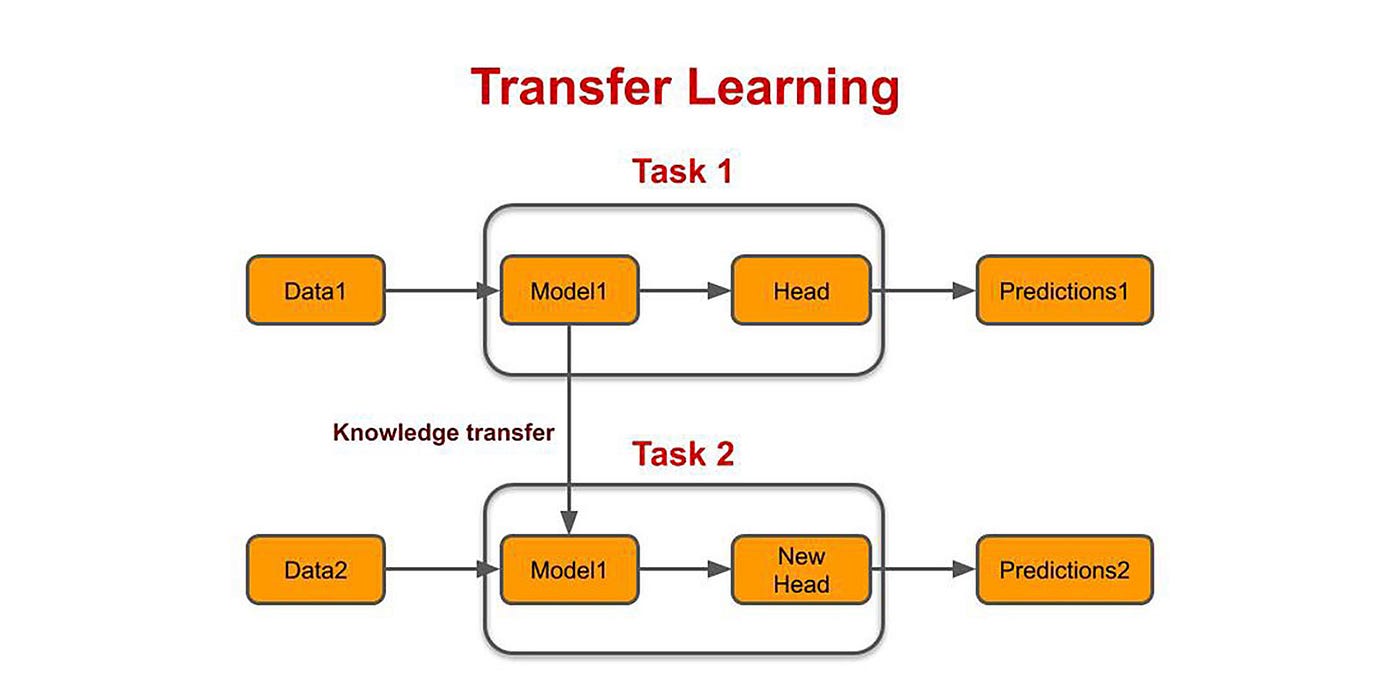
|  |  |
| --- | --- |
| **Ex No: 5**  **Date: 04/09/2024** | **Lab Record: Transfer Learning** |

**Objective:**

The objective of this lab is to classify dog breeds using the Stanford Dogs dataset and transfer learning. The MobileNetV2 model, pre-trained on ImageNet, will be used as a feature extractor to transfer its learned knowledge to this task, with minimal training on the new dataset.



**Description:**

Transfer learning involves using a pre-trained model to leverage the features learned from one task (e.g., ImageNet classification) and apply them to a different task (e.g., dog breed classification). This helps reduce training time and improves performance on the new task by transferring relevant feature representations.

**Code Line by Line Explanation:**

Necessary libraries are imported:

- `tensorflow`: Core machine learning library.

- `tensorflow\_datasets` (tfds): Used to load pre-built datasets like Stanford Dogs.

- `tensorflow\_hub`: Loads the pre-trained MobileNetV2 model.

- `Sequential` and `Dense`: Used to define the neural network layers.

- `numpy`: For numerical operations.

IMAGE\_SHAPE = (224, 224)

This sets the image shape to (224x224), the input size expected by MobileNetV2.

- tfds.load(): Loads the Stanford Dogs dataset. The dataset is split into training and test sets, and `with\_info=True` returns additional information (e.g., class labels).

- `as\_supervised=True`: Ensures the dataset is loaded as `(image, label)` pairs.

def preprocess(ds):

ds = ds.map(lambda img, label: (tf.image.resize(img, IMAGE\_SHAPE) / 255.0, label))

ds = ds.batch(32)

return ds

This function preprocesses the dataset:

- `tf.image.resize()`: Resizes images to the expected input size of 224x224.

- Normalization: The pixel values are normalized to the range [0, 1] by dividing by 255.

- `ds.batch(32)`: Organizes the dataset into batches of 32 samples.

Preprocess the training and test datasets by resizing, normalizing, and batching.

feature\_extractor\_model = "https://tfhub.dev/google/tf2-preview/mobilenet\_v2/feature\_vector/4"

pretrained\_model = hub.KerasLayer(feature\_extractor\_model, input\_shape=IMAGE\_SHAPE + (3,), trainable=False)

- `hub.KerasLayer()`: Loads the pre-trained MobileNetV2 model from TensorFlow Hub.

The model was trained on ImageNet and is used here as a feature extractor.

- `trainable=False`: This freezes the weights of the pre-trained model, preventing them from being updated during training. Only the new layers will be trained.

model = Sequential([

pretrained\_model,

Dense(dataset\_info.features['label'].num\_classes, activation='softmax')

])

- A Sequential model is defined:

- The first layer is the pre-trained MobileNetV2 used as a feature extractor.

- The second layer is a Dense layer with softmax activation, which classifies the image into one of the dog breeds (the number of classes is obtained from the dataset information).

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

The model is compiled:

- `optimizer='adam'`: Adam optimizer is used for efficient gradient descent.

- `loss='sparse\_categorical\_crossentropy'`: Loss function for multi-class classification.

- `metrics=['accuracy']`: Tracks the accuracy during training.

model.fit(train\_data, epochs=5)

Trains the model on the preprocessed training data for 5 epochs.

model.evaluate(test\_data)

Evaluates the model's performance on the test dataset, calculating accuracy and loss.

`matplotlib.pyplot` is imported to display the image.

def preprocess\_image(image\_path):

img = tf.io.read\_file(image\_path)

img = tf.image.decode\_jpeg(img, channels=3)

img = tf.image.resize(img, IMAGE\_SHAPE)

img = img / 255.0

return tf.expand\_dims(img, 0)

This function preprocesses a single image:

- `tf.io.read\_file(image\_path)`: Reads the image file.

- `tf.image.decode\_jpeg()`: Decodes the image as a JPEG.

- `tf.image.resize()`: Resizes the image to 224x224.

- Normalization: Pixel values are scaled to [0, 1].

- `tf.expand\_dims()`: Adds a batch dimension since the model expects batched input.

test\_image\_path = 'dog.jpg'

test\_image = preprocess\_image(test\_image\_path)

Loads and preprocesses the test image.

predictions = model.predict(test\_image)

predicted\_class = np.argmax(predictions, axis=1)

Predicts the class of the dog breed from the test image:

- `model.predict()`: Makes predictions using the trained model.

- `np.argmax()`: Gets the index of the class with the highest prediction probability.

predicted\_label = dataset\_info.features['label'].int2str(predicted\_class[0])

Converts the predicted class index into a human-readable label.

plt.figure(figsize=(6, 6))

img = tf.io.read\_file(test\_image\_path)

img = tf.image.decode\_jpeg(img, channels=3)

img = tf.image.resize(img, IMAGE\_SHAPE)

img = img / 255.0

plt.imshow(img)

plt.title(f"Predicted dog breed: {predicted\_label}")

plt.axis('off')

plt.show()

Displays the test image with its predicted label:

- `plt.imshow()`: Displays the image.

- `plt.title()`: Adds the predicted label as the title of the plot.

This implementation demonstrates transfer learning by leveraging the pre-trained MobileNetV2 model. Instead of training from scratch, the pre-trained model extracts features from images, which are passed to a classification layer to predict the dog breed. This significantly reduces the training time and enhances performance, especially with a smaller dataset like Stanford Dogs.

**GitHub Link:** https://github.com/dyuthiramesh/Deep\_Learning\_Elective/blob/main/Sem5/Lab5/cnn\_transfer\_learning\_DISTRI.ipynb