Latihan Soal
Exercise

Card Classification using Deep Convolutional Neural Network

As the latest recruit, you are tasked with spearheading this groundbreaking project. Your mission is to develop a Python-based program utilizing the TensorFlow library to craft a Deep Convolutional Neural Network (DCNN) model. This advanced model will specialize in accurately distinguishing between **four primary card classes**: Ace of Diamonds, Five of Spades, Jack of Hearts, and Six of Diamonds. Your task as a deep learning engineer is to develop a Python-based program using the **TensorFlow library** to create a **Deep Convolutional Neural Network (DCNN) model** that meets the following requirements:

- **Load and Split Dataset**
  - **Load Dataset**

    In this section, you are required to:
    - **Assign** the dataset path to the previously declared `PATH` variable.
    - **Populate** the `CLASS_NAMES` variable with the appropriate class names based on the provided information.
    - **Load** the image from the given dataset into an array using grayscale as the color mode.
    - **Randomly** shuffle the dataset.
  - **Split Dataset**

    You are tasked to **split** the given dataset into 3 main parts which are training, validation, and testing sets with the proportion of 70/15/15.
- **Data Preprocessing**

  You are tasked with **reshaping** the given dataset, which includes the training, validation, and testing sets, by converting the data type to `float32`. Following this, you will need to **normalize** the dataset, comprising the training, validation, and testing sets, by dividing the values of all image pixels by `255`. As an optional but potentially beneficial step, consider utilizing **categorical encoding** for the target variables in the training, validation, and testing sets.

- **DCNN Architecture (Model Architecture Visualization, Initialization, and Configuration)**
  - **Model Visualization**

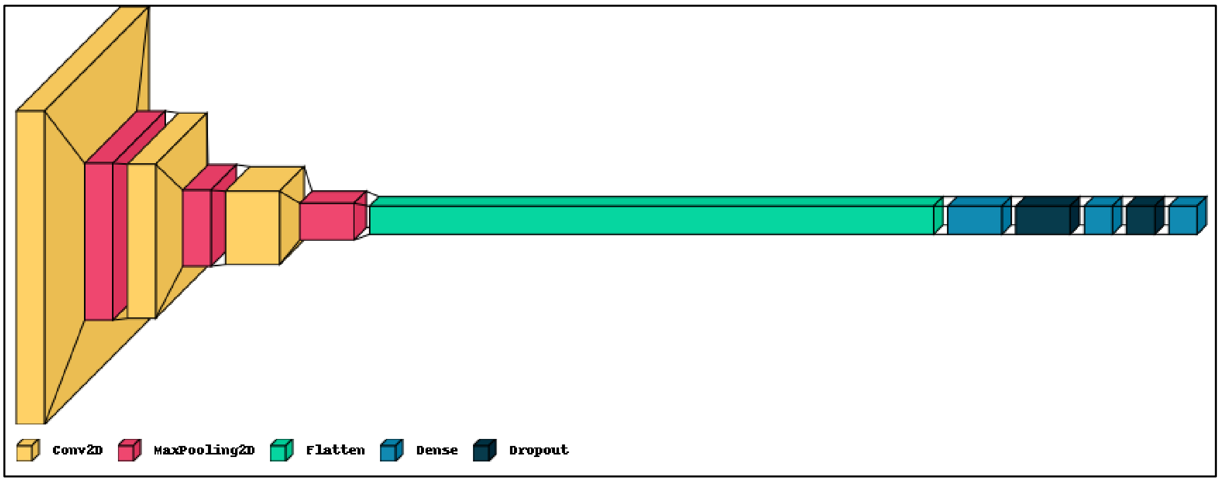

Figure 1. Model Visualization

  - **Model Initialization and Configuration – Input Layer**

    In this section, you are tasked with initializing the **input layer** of the model. You need to define the dimensions of the image and the channel to be processed by the input layer.

  - **Model Initialization and Configuration – Hidden Layer**

    In this section, your responsibility is to detail or outline the configurations of the hidden layers of the CNN model. The architecture starts with an Input layer followed by a sequence of **convolutional layers** and **max pooling layers** that function to process the input data and reduce feature map dimensions, respectively. Subsequently, a flatten layer is integrated, facilitating the transition from multi-dimensional to one-dimensional data before proceeding to the fully connected layers. A **dropout layer** is incorporated within the structure to mitigate overfitting during the training phase. **Provide** a **rationale** for selecting the specific activation function and **describe** the function of each layer mentioned.

  - **Model Initialization and Configuration – Output Layer**

    In this section, you are tasked with describing the output layer of the DCNN model. The model culminates in a dense layer with 4 units. This final **dense layer** employs a `Softmax` activation function which allows the model to classify input images into one of the four possible categories.

- **Model Training**

  You are tasked with **training the model** that you previously constructed. **Utilize** the Adam optimizer and **select** the appropriate loss function based on the specifics of your case. **Monitor** the performance using accuracy as the metric. Ensure that the training process spans a minimum of 40 epochs. You have the flexibility to set the batch size and learning rate for the optimizer according to your judgment.

```
Epoch 30/40
7/7 [==============================] - 1s 86ms/step - loss: 0.5032 - accuracy: 0.8232 - val_loss: 0.4548 - val_accuracy: 0.8315
Epoch 31/40
7/7 [==============================] - 1s 83ms/step - loss: 0.5237 - accuracy: 0.7966 - val_loss: 0.4606 - val_accuracy: 0.8315
Epoch 32/40
7/7 [==============================] - 1s 82ms/step - loss: 0.5448 - accuracy: 0.8063 - val_loss: 0.4564 - val_accuracy: 0.8427
Epoch 33/40
7/7 [==============================] - 1s 82ms/step - loss: 0.4853 - accuracy: 0.8354 - val_loss: 0.4979 - val_accuracy: 0.8315
Epoch 34/40
7/7 [==============================] - 1s 86ms/step - loss: 0.4895 - accuracy: 0.8160 - val_loss: 0.4618 - val_accuracy: 0.8427
Epoch 35/40
7/7 [==============================] - 1s 85ms/step - loss: 0.4270 - accuracy: 0.8426 - val_loss: 0.4705 - val_accuracy: 0.8539
Epoch 36/40
7/7 [==============================] - 1s 84ms/step - loss: 0.4494 - accuracy: 0.8378 - val_loss: 0.4875 - val_accuracy: 0.8427
Epoch 37/40
7/7 [==============================] - 1s 84ms/step - loss: 0.4426 - accuracy: 0.8475 - val_loss: 0.4934 - val_accuracy: 0.8427
Epoch 38/40
7/7 [==============================] - 1s 83ms/step - loss: 0.4111 - accuracy: 0.8426 - val_loss: 0.4123 - val_accuracy: 0.8764
Epoch 39/40
7/7 [==============================] - 1s 80ms/step - loss: 0.3953 - accuracy: 0.8571 - val_loss: 0.4507 - val_accuracy: 0.8539
Epoch 40/40
7/7 [==============================] - 1s 83ms/step - loss: 0.3974 - accuracy: 0.8450 - val_loss: 0.4330 - val_accuracy: 0.8539
```

Figure 2. Model Training

- **Predicting Data Using Created Model and Model Evaluation**
  - **Predict Data**

    You are tasked with **predicting** the outcomes of the testing sets using the model that you previously constructed. Please print out both the loss and accuracy values.

```
3/3 [==============================] - 0s 9ms/step
3/3 [==============================] - 0s 13ms/step - loss: 0.5854 - accuracy: 0.8090
```

Figure 3. Model Prediction

  - **Model Evaluation**

    You are tasked with **evaluating** the testing sets using the model that you previously constructed. Please print out both the loss and accuracy values.

```
Test Accuracy :  0.8089887499809265
Test Loss     :  0.585367739200592
```

Figure 4. Model Evaluation

- **Plot of Evaluation Metrics**

  You are tasked with creating a **plot** of the evaluation metrics. This plot should include a loss graph, which displays both the training and validation losses, as well as an accuracy graph that illustrates the training and validation accuracies.

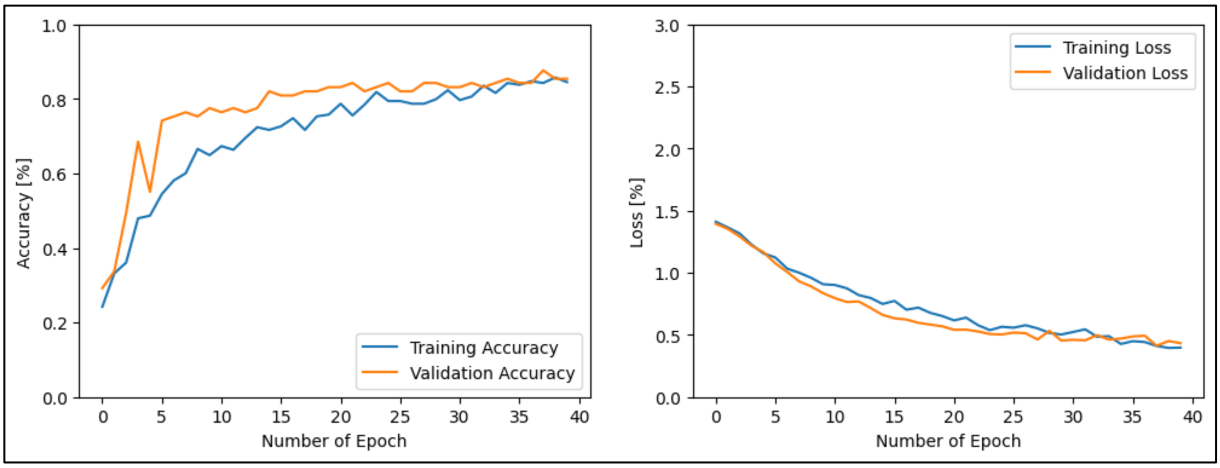

Figure 5. Plot of Evaluation Metrics