

**Artificial Intelligence**

**Module 5 Assignment**

[support@intellipaat.com](mailto:support@intellipaat.com)

+91-7022374614

US: 1-800-216-8930 (Toll-Free)

**Assignment Questions**

.

**Tasks to be performed:**

1. Briefly describe the disadvantages of Flattening (as first layer or after Input) and how do you overcome those issues ?

*Flattening is a process of converting multidimensional data into a one-dimensional vector. This operation is commonly used as the first layer or after the input layer in deep learning models. While flattening can simplify the input data for the model, it has some drawbacks:*

* 1. *Loss of Spatial Information: Flattening can lead to the loss of spatial information in the input data, such as the spatial relationship between pixels in an image. This can limit the model's ability to capture complex patterns in the data.*
  2. *Increased Dimensionality: Flattening can also lead to an increase in dimensionality of the input data, which can make the model more complex and difficult to train. This can also result in the "curse of dimensionality" problem, where the amount of data required to accurately represent the input space grows exponentially with the number of dimensions.*

*To overcome these issues, several techniques can be used, such as:*

* 1. *Using convolutional layers instead of flattening: Convolutional layers are designed to preserve spatial information in the input data and can be used to extract features from multidimensional data, such as images or time-series data.*
  2. *Dimensionality Reduction Techniques: Instead of flattening, we can use techniques like PCA, LDA, t-SNE to reduce the dimensionality of input data.*
  3. *Regularization Techniques: Using techniques like dropout regularization or early stopping to prevent overfitting and reduce the complexity of the model.*
  4. *Using Transfer Learning: Transfer learning is a technique where a pre-trained model is used as a starting point for a new model, instead of training from scratch. This can help to reduce the complexity of the model and improve its performance.*

1. Stanford Vision department, has collected and annotated images of 120 breeds of dogs from ImageNet. [Dataset can be found here](https://www.kaggle.com/c/dog-breed-identification).
2. Prepare an EDA sheet & explain what you understand from this data. What would be your approach?

*The competition provide a label.csv file having 10222 rows and 2 columns with id and breed name.The images are saved in train folder.there is 120 breeds in the data set.The images in the train folder is having different image size.Majority of the images having width in range of 300-600 and height 400-700.We can consider the input image as 150,150*

import numpy as np

import pandas as pd

import os,cv2

import matplotlib.pyplot as plt

labels=pd.read\_csv('labels.csv')

labels.head()

labels['filename']=labels['id']+'.jpg'

labels.head()

|  | **id** | **breed** | **filename** |
| --- | --- | --- | --- |
| **0** | 000bec180eb18c7604dcecc8fe0dba07 | boston\_bull | 000bec180eb18c7604dcecc8fe0dba07.jpg |
| **1** | 001513dfcb2ffafc82cccf4d8bbaba97 | dingo | 001513dfcb2ffafc82cccf4d8bbaba97.jpg |
| **2** | 001cdf01b096e06d78e9e5112d419397 | pekinese | 001cdf01b096e06d78e9e5112d419397.jpg |
| **3** | 00214f311d5d2247d5dfe4fe24b2303d | bluetick | 00214f311d5d2247d5dfe4fe24b2303d.jpg |
| **4** | 0021f9ceb3235effd7fcde7f7538ed62 | golden\_retriever | 0021f9ceb3235effd7fcde7f7538ed62.jpg |

print('Number train samples:', len(labels))

print('Number train images:', len(os.listdir('train/')))

Number train samples: 10222 Number train images: 10222

labels.groupby('breed').count()['filename']

There are 120 breads

1. With a limited number of training images per class, what extra steps will you incorporate for a good model?

*Data augmentation: By generating additional training data from the existing images, we can increase the number of examples per class and improve the robustness of the model. Techniques such as random cropping, flipping, rotating, and adding noise can be used to create new images.*

idg = tf.keras.preprocessing.image.ImageDataGenerator(validation\_split=0.1, rotation\_range=30, horizontal\_flip=True)

*Transfer learning: We can leverage pre-trained models that have been trained on large datasets to improve the performance of our model on the limited dataset. By fine-tuning a pre-trained model on our dataset, we can improve its ability to recognize the features specific to our dataset.*

vgg\_model = tf.keras.applications.VGG16()

vgg\_model.summary()

for layer in vgg\_model.layers:

  layer.trainable = False

vgg\_model.layers[-3].name # i want the output of 3rd last layer which is the Dense layer

vgg\_out = vgg\_model.layers[-3].output

bn = tf.keras.layers.BatchNormalization(trainable=False) (vgg\_out)

d1 = tf.keras.layers.Dense(1024, activation='relu') (bn)

dp1 = tf.keras.layers.Dropout(0.5) (d1)

output = tf.keras.layers.Dense(120, activation='softmax') (dp1)

final\_model = tf.keras.models.Model(inputs=[vgg\_model.input], outputs=[output])

final\_model.summary()

final\_model.compile(tf.keras.optimizers.SGD(learning\_rate=0.005),

                    loss=tf.keras.losses.categorical\_crossentropy,

                    metrics=["acc"])

final\_model\_his = final\_model.fit(train\_idg, epochs=30, validation\_data=val\_idg)

1. Achieve the above problem on Kaggle itself and submit your results.

Please check [notebook38461769b4 | Kaggle](https://www.kaggle.com/code/jijinakv/notebook38461769b4?scriptVersionId=120396580)

1. What are Convolutional Neural Networks and how are they better than regular only Fully Connected Neural Networks?

*Convolutional Neural Networks (CNNs) are a type of neural network architecture designed for image processing tasks. CNNs are based on the idea of applying a set of learned filters to the input image, extracting local features in a hierarchical manner, and using these features to classify the image.*

*CNNs are better than regular fully connected neural networks for image processing tasks for several reasons:*

*Parameter Sharing: CNNs use a shared weight scheme, where the same set of weights are used for each local region of the input image. This dramatically reduces the number of parameters required to train the network, making it more efficient and easier to optimize.*

*Translation Invariance: The learned filters in a CNN are capable of detecting features regardless of their position in the input image. This property is known as translation invariance, and it allows CNNs to recognize patterns in an image regardless of their location, making them well-suited for image classification tasks.*

*Pooling Layers: CNNs often use pooling layers to downsample the output of the convolutional layers, reducing the dimensionality of the data and increasing the model's ability to recognize patterns at different scales.*

*Hierarchical Feature Learning: CNNs are designed to extract features from an input image in a hierarchical manner, starting from low-level features such as edges and textures and building up to more complex features such as object parts and shapes. This makes CNNs particularly effective for recognizing objects in images.*

*In summary, CNNs are better suited for image processing tasks than regular fully connected neural networks because of their ability to share parameters, achieve translation invariance, use pooling layers for dimensionality reduction, and extract hierarchical features.*

1. The major components of a CNN topology are Filter/ Kernel, Strides, etc. Explain mathematically the output of a single CNN layer and how it is impacted by strides?

*The output of a single CNN layer can be computed mathematically using the following formula:*

*output\_size = ((input\_size - filter\_size)/stride) + 1*

*Where:*

*input\_size: the size of the input volume (width, height, and depth)*

*filter\_size: the size of the filter/kernel (width, height, and depth)*

*stride: the number of pixels that the filter/kernel moves in each step*

*The output size is calculated separately for each dimension of the input volume, and the resulting output volume has the same depth as the number of filters used in the layer. The output size is also impacted by the padding used in the layer, which can be either "valid" (no padding) or "same" (padding is added to the input volume to ensure that the output size is the same as the input size).*

*The stride parameter controls the step size of the filter/kernel as it moves across the input volume. A larger stride results in a smaller output size, as the filter/kernel covers a smaller portion of the input volume in each step. Conversely, a smaller stride results in a larger output size, as the filter/kernel covers a larger portion of the input volume in each step.*

*In summary, the output of a single CNN layer is computed based on the input size, filter size, stride, and padding used in the layer. The stride parameter impacts the size of the output volume, with a larger stride resulting in a smaller output size and a smaller stride resulting in a larger output size.*

1. For a single channel image, size (9x9) and a filter of size (2x2) and stride (2x2), calculate the output. Is padding required ? What is the output size of this convoluted feature?

*Given:*

*Input image size: 9x9*

*Filter/kernel size: 2x2*

*Stride: 2x2*

*We can calculate the output size using the formula:*

*output\_size = ((input\_size - filter\_size)/stride) + 1*

*where input\_size = 9, filter\_size = 2, and stride = 2.*

*Substituting the values:*

*output\_size = ((9 - 2) / 2) + 1 output\_size = 4*

*Therefore, the output feature map will have a size of 4x4.*

*Padding is not required in this case, as the input size is already divisible by the stride size without any remainder. If padding were required, it would be added to the input image to ensure that the output size is the same as expected. Padding can be added as "valid" (no padding) or "same" (adding padding to the input image to ensure the output size is the same).*

1. For a particular problem if the error is not dipping below a standard point, what are the techniques used to reduce error ? Explain the methodology in depth and with examples.

*if a deep learning model for image classification is not improving beyond a certain point, one could try data augmentation to create new training examples, or use transfer learning by fine-tuning the weights of a pre-trained model. Alternatively, one could try adding dropout to the network or tuning the learning rate to improve the model's generalization ability. In general, it is recommended to start with simpler techniques such as data preprocessing and hyperparameter tuning before moving on to more advanced techniques such as ensemble methods or architecture design.*