Goal: Learning transfer learning on CNNs - ResNet, VGG16, InceptionNet

- Libraries → numpy, Pandas, tensorflow.keras.preprocessing.image → load_img, img_to_array

- Step 1: Preparing the data: load the Image using load_img & Reshape to 150×150

```
img = load_img(img_path, target_size=(150, 150))
```

, next convert it to array & Normalize and add to our traing set

Do the same for all types of Classes with their labels

- Step 2: Concatenate different classes into x_train & their labels into y_train

we get (14034, 150, 150, 3) & (14034,)

Case I. Import Inception model from tensorflow.keras.applications.inception_v3 import InceptionV3

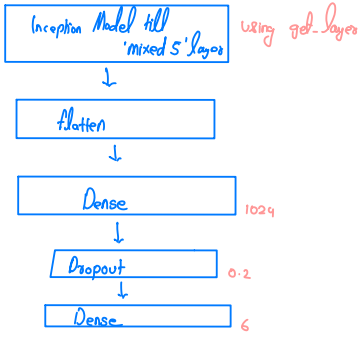
To load weights from a local file:

```
pre_trained_model = InceptionV3(input_shape = (150,150,3), include_top=False, weights=None)
pre_trained_model.load_weights('location_of_weights_for_inception_model')
```

- In my case I have trained of weight training

i.e. layers_name.trainable = false

Making our own model on top of Inception_V2



- Compile with → optimizer = RMSprop
  → loss = 'sparse_categorical_crossentropy'
  → metrics = Accuracy
- fit → epoch = 10

→ Other case could be we use weights of Imagenet, which are inbuilt parameter of Inception_V2

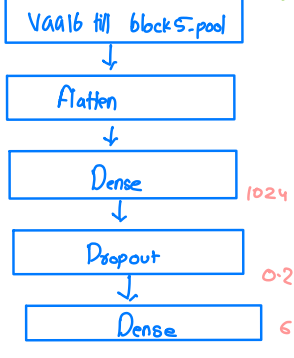
CASE II: VGG16

(A) VGG16 with Imagenet weights

VGG16 can be made using VGG16 as

pretrained_model = VGG16 (input_shape = (150, 150, 3), include_top = False, weights = 'imagenet')

Get the



Case III: ResNet 50

· ResNet with inbuilt pretrained weights by ImageNet

pretrained_model = ResNet50 (input_data = (150, 150, 3) , include_top = False, weights = 'imagenet')

we can modify it as