Skin Cancer Detection

By Pranav Reddy Gunda , Nihaal Chowdary Surpani , Sharath Sen Reddy Vellanki , Akash Parmar

INTRODUCTION

- **Dermatoscopy and Neural Networks:** Dermatoscopy is a technique that helps doctors examine skin lesions more closely. Images taken this way are useful for training computers to automatically diagnose different skin conditions. Back in 1994, researchers successfully used dermatoscopic images to teach a computer to tell apart melanomas (a serious skin cancer) from common moles. However, at that time, the study had a small number of images and focused mainly on melanomas and moles.
- **Challenges in Training Computers:** Nowadays, thanks to better technology and machine learning, we expect computers to become really good at diagnosing various skin lesions. But there's a challenge - training these computer programs requires a large number of pictures of skin issues with clear diagnoses. Unfortunately, there aren't many high-quality images available for all types of skin diseases.
- **HAM10000 Dataset:** Researchers are working on teaching computers to recognize different skin issues using special images. They faced challenges because there weren't enough diverse pictures for training. Existing datasets had limitations, mainly focusing on certain types of skin problems. To overcome this, they released a new dataset called HAM10000 ("Human Against Machine with 10000 training images"), aiming to help computers become better at diagnosing various skin conditions and comparing their performance with human experts.

HAM10000 Dataset

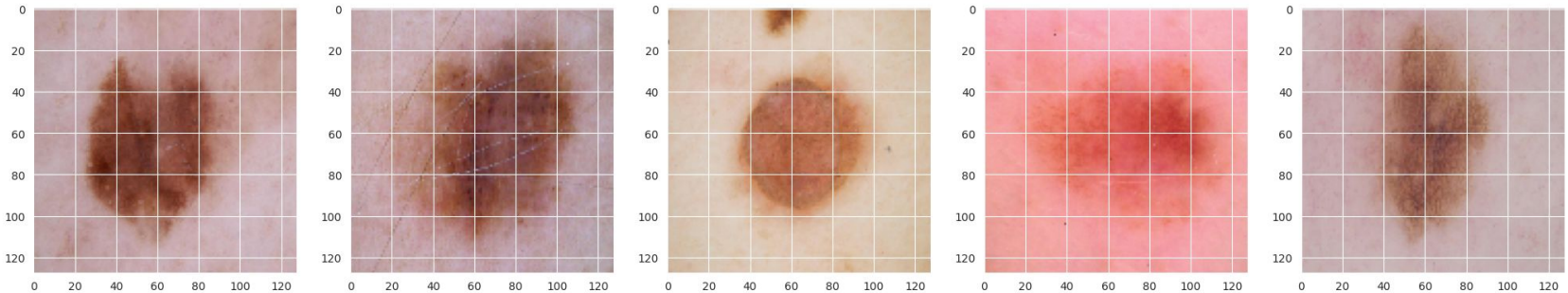
The HAM10000 training set includes pigmented lesions from different populations. We use TensorFlow and Keras to implement a Convolutional Neural Network (CNN) using the ResNet50 architecture for image classification. There are 7 columns in the Dataset:

1. lesion_id: A unique identifier for each skin lesion.
2. image_id: Identifier for the corresponding image of the lesion.
3. dx: Diagnosis category for the skin lesion (e.g., nv for nevi, mel for melanoma).
4. dx_type: Type of diagnosis method used (e.g., histopathology, follow-up).
5. age: Age of the individual with the skin lesion.
6. sex: Gender of the individual (male or female).
7. localization: The location on the body where the lesion is found.

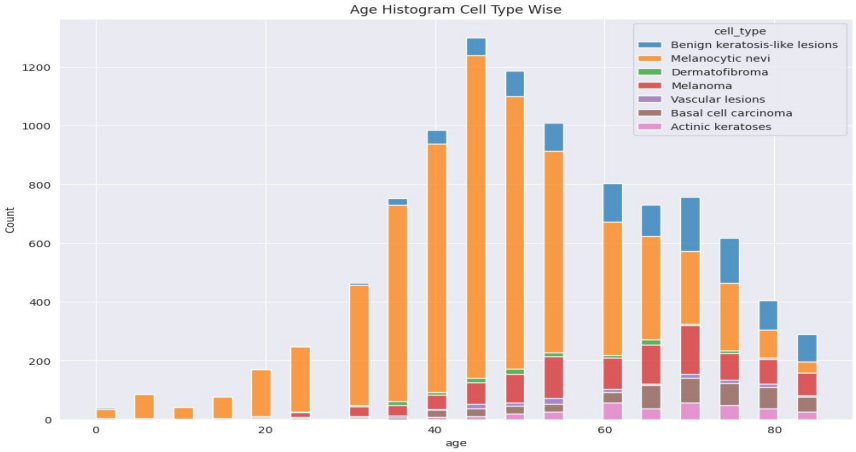
DATA

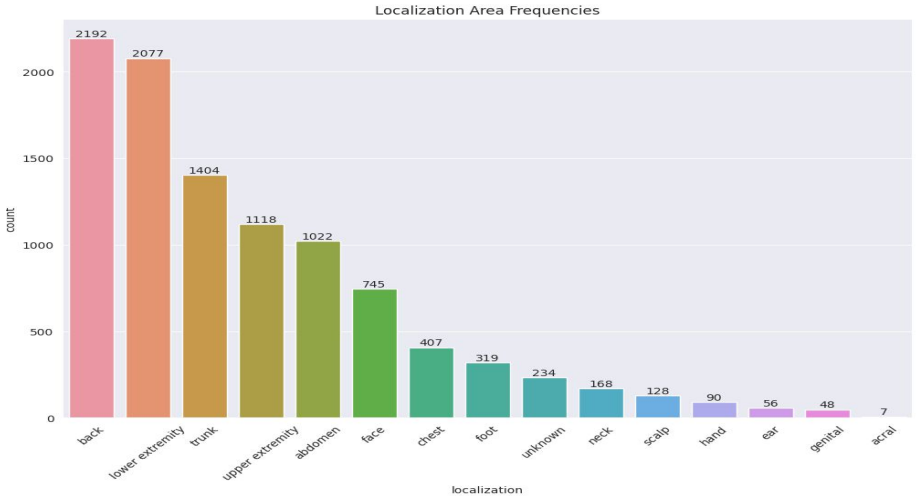
10015 Images of 7 different types of skin diseases

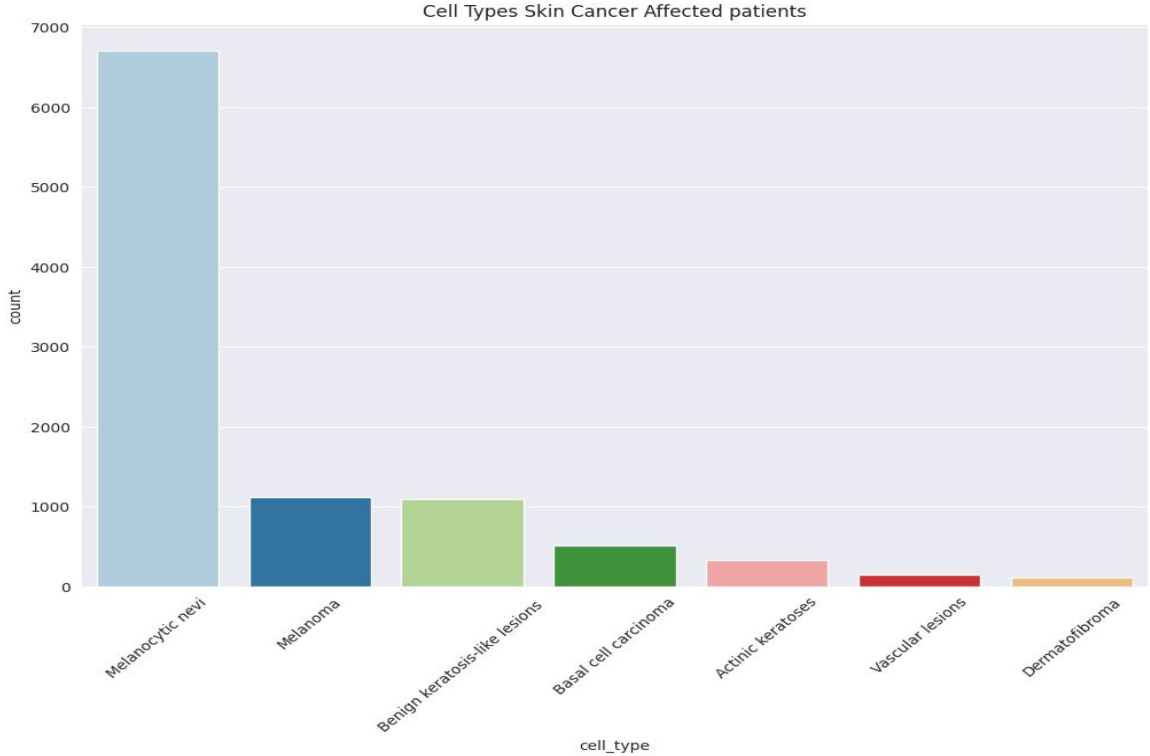
- 'Melanocytic nevi'
- 'Melanoma'
- 'Benign keratosis-like lesions '
- 'Basal cell carcinoma'
- 'Actinic keratoses'
- 'Vascular lesions'
- 'Dermatofibroma'



Analysis


Age Histogram Cell Type Wise
cell_type
Benign keratosis-like lesions
Melanocytic nevi
Dermatofibroma
Melanoma
Vascular lesions
Basal cell carcinoma
Actinic keratoses
Count
age
0
200
400
600
800
1000
1200
0
20
40
60
80


Localization Area Frequencies
count
localization
2192
2077
1404
1118
1022
745
407
319
234
168
128
90
56
48
7
back
lower extremity
trunk
upper extremity
abdomen
face
chest
foot
unknown
neck
scalp
hand
ear
genital
acral
0
500
1000
1500
2000


Cell Types Skin Cancer Affected patients
7000
6000
5000
4000
3000
2000
1000
0
count
Melanocytic nevi
Melanoma
Benign keratosis-like lesions
Basal cell carcinoma
Actinic keratoses
Vascular lesions
Dermatofibroma
cell_type

Data Augmentation

Addressing Data Inconsistency

```
Class 0: 327 occurrences
Class 1: 514 occurrences
Class 2: 1099 occurrences
Class 3: 115 occurrences
Class 4: 1113 occurrences
Class 5: 6705 occurrences
Class 6: 142 occurrences
```

Data Augmentation

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

```
Class 0.0: 500 occurrences
Class 1.0: 500 occurrences
Class 2.0: 500 occurrences
Class 3.0: 500 occurrences
Class 4.0: 500 occurrences
Class 5.0: 500 occurrences
Class 6.0: 500 occurrences
```

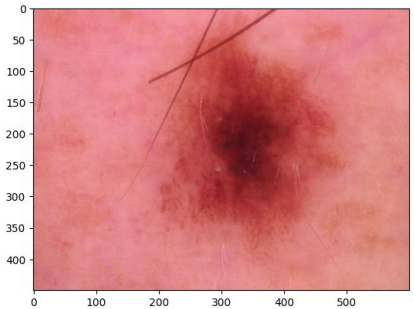
- Considered 500 images for each class due to computational complexity

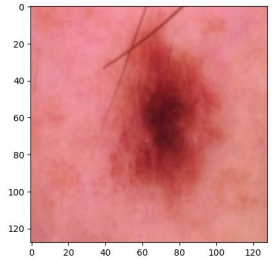
Training and Testing data

The data is splitted into Training, Validation and Testing sets

- 70% Training set (2450 images)
- 15% validation set (525 images)
- 15% Testing set.(525 images)

The images are resized from (400 * 600 * 3) to (128 * 128 * 3) due to computational complexity.



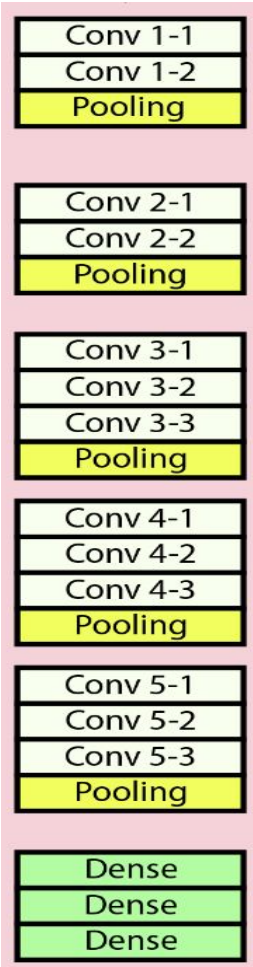


Transfer Learning

- **Leverage Pre-trained Models**: Utilize pre-trained models, like InceptionV3, which have been trained on large datasets to extract valuable features from images.

- **Feature Extraction**: Extract important features, such as edges and patterns, from the images using the pre-trained models feature extraction layers.

- **Fine-tuning**: Fine-tune the final layers of the pre-trained model by adjusting the weights to adapt it to the specific classification task, such as skin cancer classification.

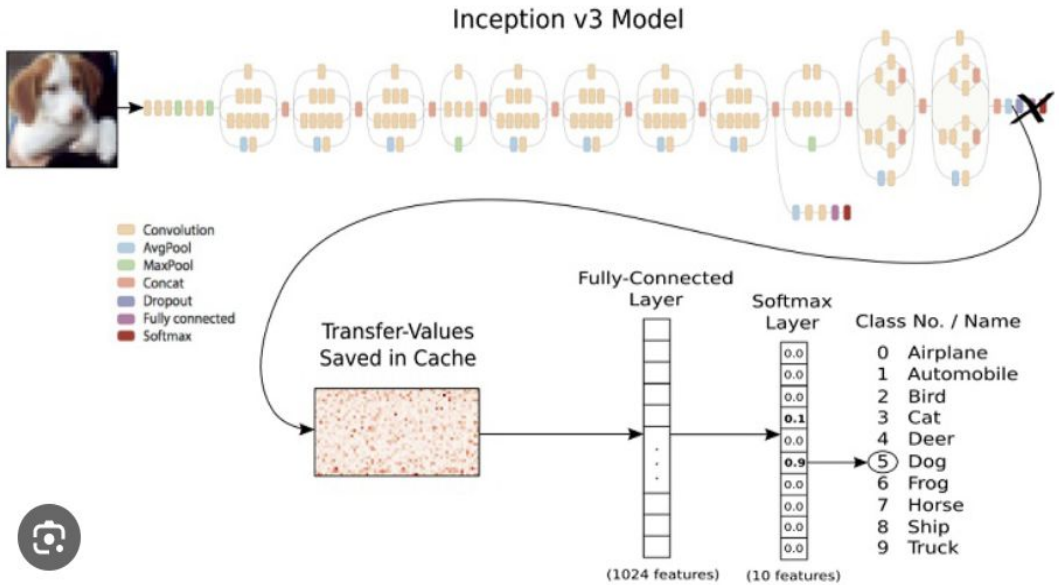


Inception V3

- Inception v3, an influential convolutional neural network (CNN), was developed by a team of researchers at Google led by Christian Szegedy. Introduced in 2015, it represents a significant advancement in Deep Learning.

- Inception v3's architecture comprises multiple sophisticated layers, including convolutional and pooling layers, such as 1x1, 3x3, and 5x5 convolutions, and pooling, enabling the network to capture diverse features across different spatial scales within an image.

- Inception v3 finds widespread usage in computer vision applications, including image classification, object detection, and image segmentation. Its efficient design, which balances performance and computational resources, makes it a popular choice in various industries.

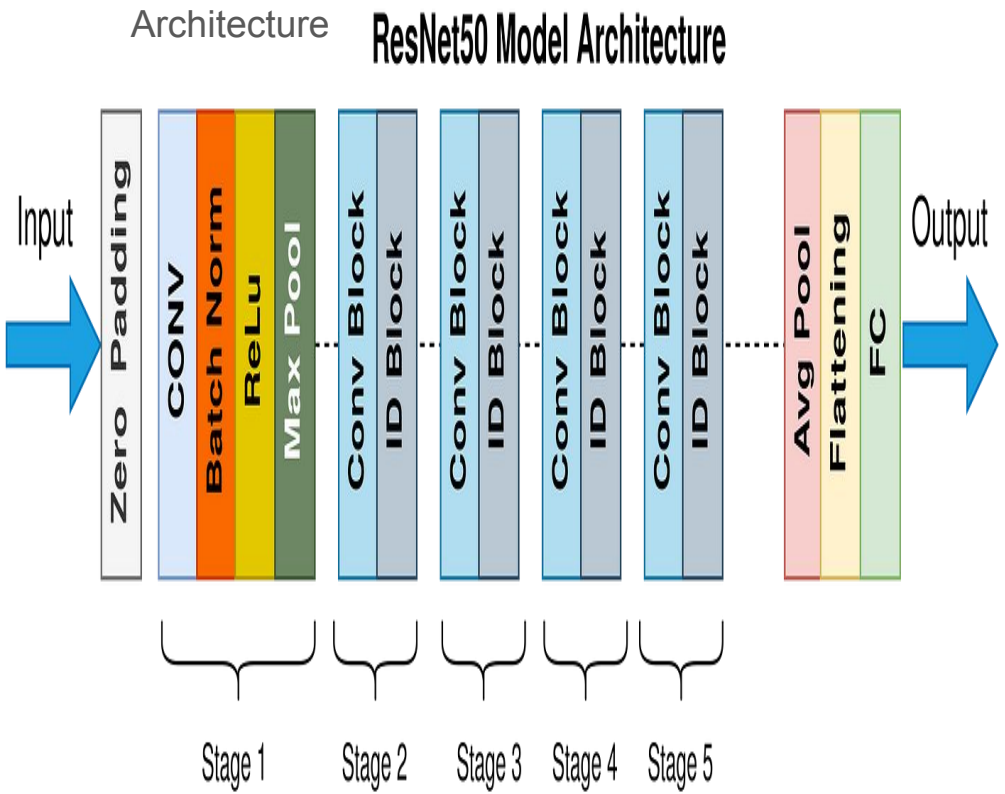


Code

```
inception = InceptionV3(input_shape=(128,128,3), weights='imagenet', include_top=False)
# don't train existing weights
for layer in inception.layers:
    layer.trainable = False

# our layers - you can add more if you want
x = Flatten()(inception.output)
prediction = Dense(y_train_array.shape[1] , activation='softmax')(x)

# create a model object
model = Model(inputs=inception.input, outputs=prediction)
```

ResNet-50

- It was first introduced in Dec 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun @ MSRA(Microsoft Asia). The suffix 50 states the number of layers used in architecture.
- Need for Resnet :To overcome the problem of vanishing gradient. The gradients become very small as they are propagated through many layers of the neural network.
- As the number of layers increases so does the training and testing errors which can lead to slower learning or complete lack of learning.Uses skip connections to bypass certain layers in network.
- ResNet contains 1 convolutional layer in stage1 along with batch normalization , ReLU function and max pooling. There are 9 convolutional layers in stage 2 there are 12 convolutional layers in stage 3 , 18 convolutional layers in stage 4 , 9 convolutional layers in stage 5 followed by batch normalization and ReLU activation function. Stage 5 contains a average pooling which reduces dimensions of output tensor to a vector.



ResNet-50

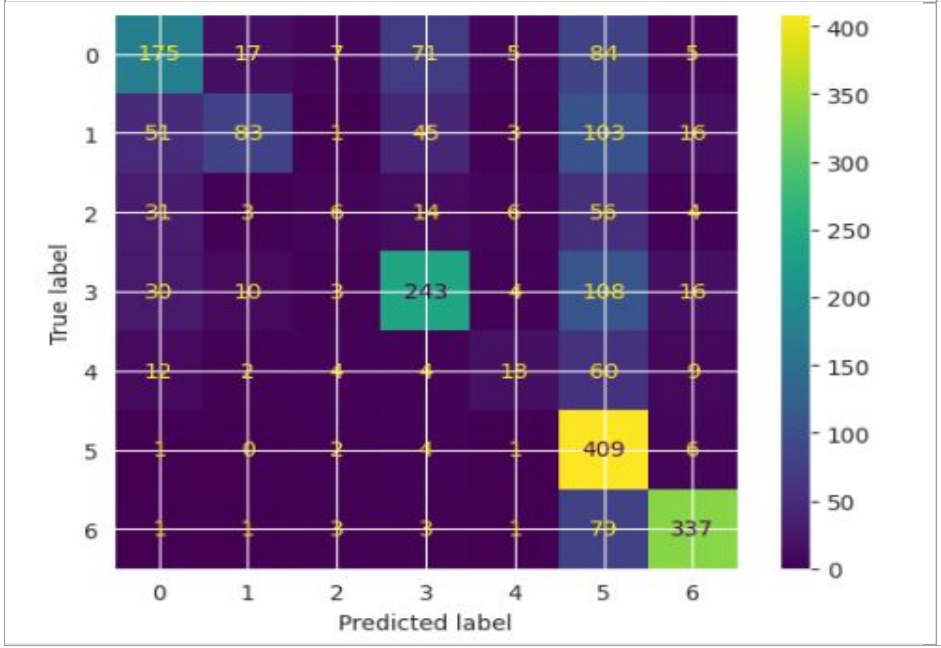
```
resnet = ResNet50(input_shape= (128,128,3), weights='imagenet', include_top=False)

# don't train existing weights
for layer in resnet.layers:
    layer.trainable = False

# our layers - you can add more if you want
x = Flatten()(resnet.output)
prediction = Dense(y_train_array.shape[1], activation='softmax')(x)


# create a model object
model2 = Model(inputs=resnet.input, outputs=prediction)
```

Results

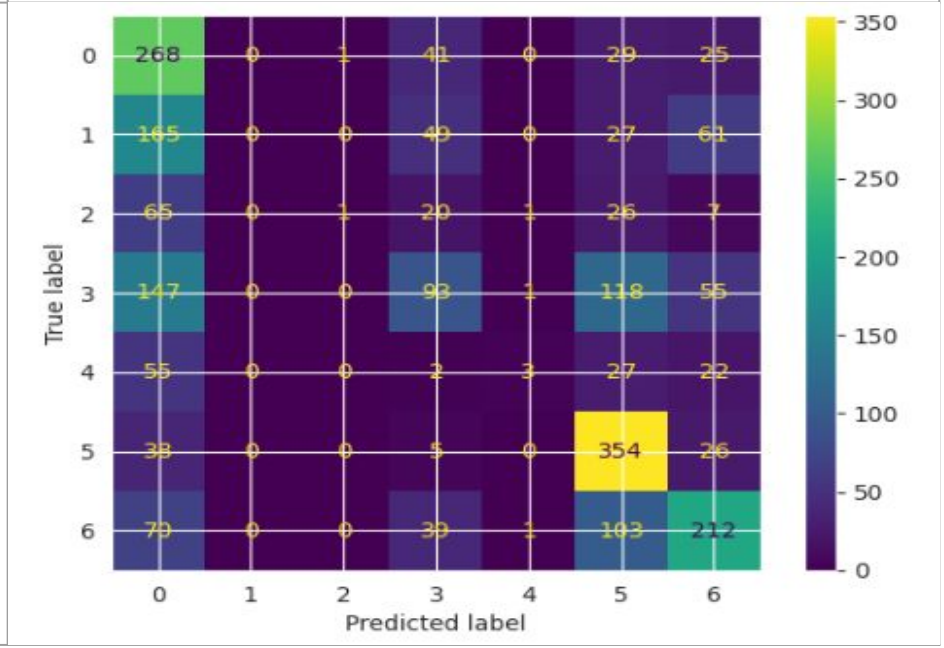
Accuracy Scores

Inception_v3	0.5892443208159481
ResNet-50	0.5016179879462216

Inception_v3

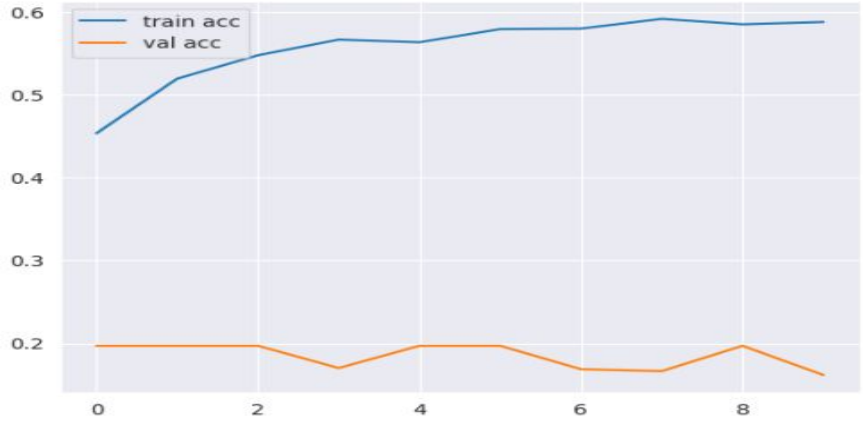


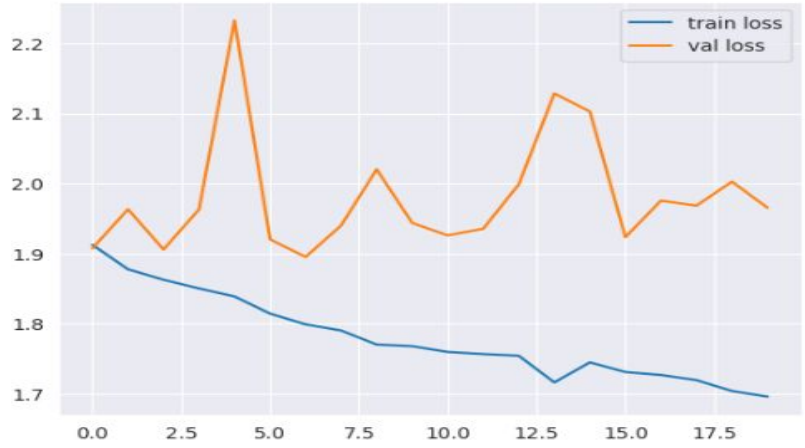
ResNet-50

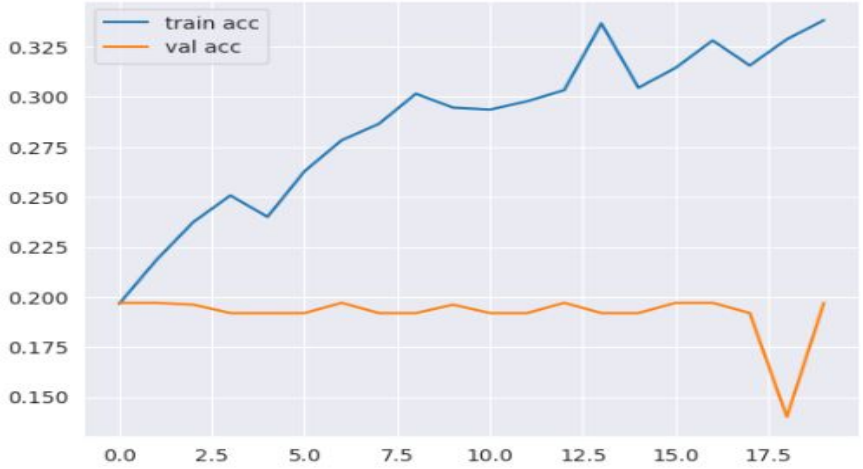


Results


train loss
val loss
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
0
2
4
6
8


train acc
val acc
0.6
0.5
0.4
0.3
0.2
0
2
4
6
8


train loss
val loss
2.2
2.1
2.0
1.9
1.8
1.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


train acc
val acc
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5