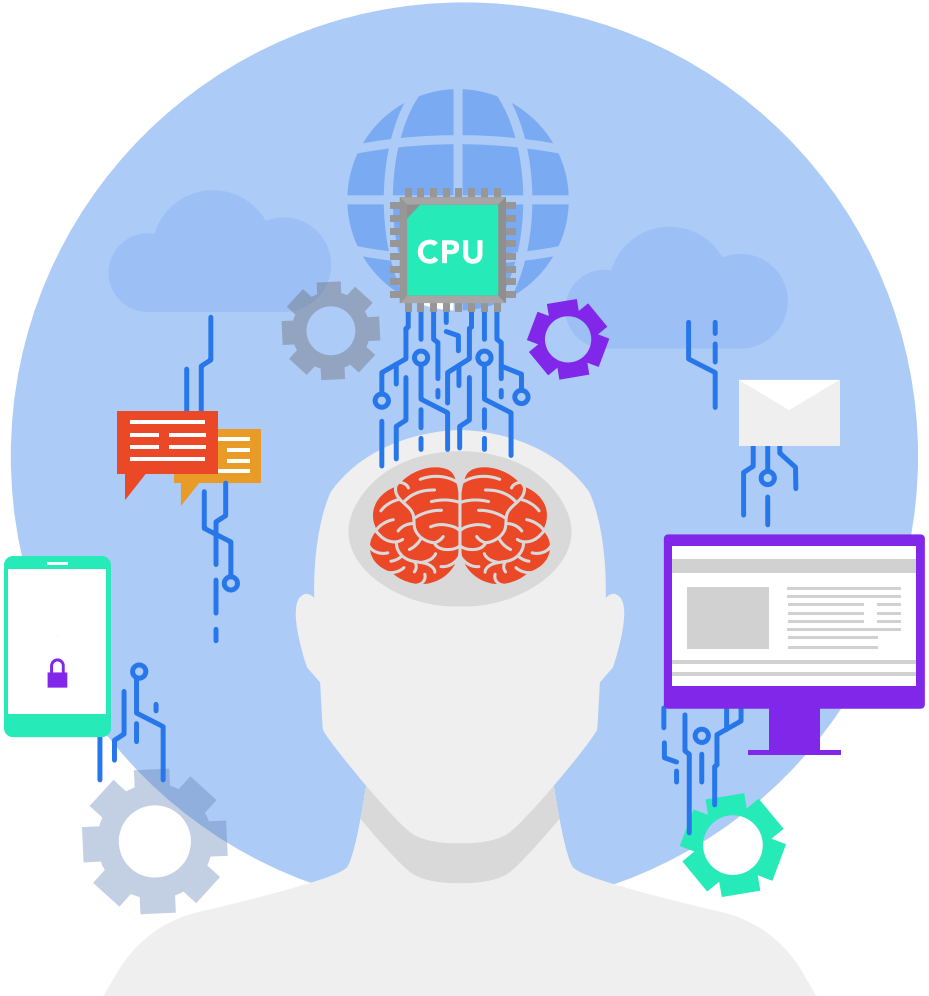
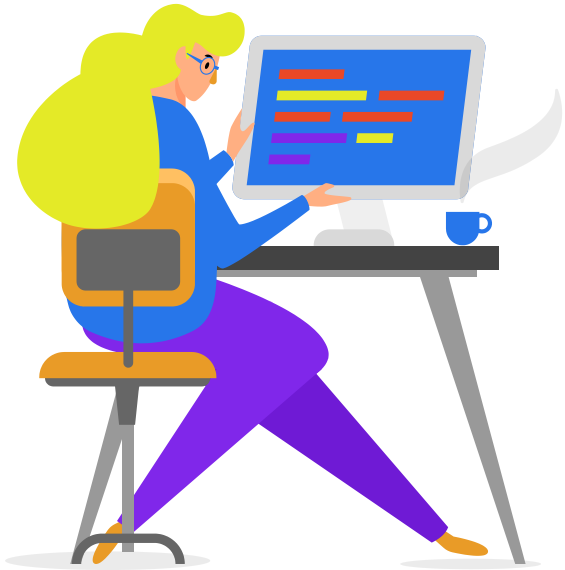


Kaggle - Plant Seedling Classification

Tran Hien Van - U1920891J
Agrawal Rachita - U1922919L
Tayal Aks - U1922309J
Acharya Atul - U1923502C

Agenda



01 **Problem Statement**

02 **Exploratory Data Analysis**

Overview, Data distribution, PCA

03 **Data Preprocessing**

Normalization, Balancing, Image Segmentation

04 **Solutions Implemented**

KNN, K-Means, CNN, Transfer Learning, Vision Transformers, Other solutions

05 **Leaderboard**

Final Ranking

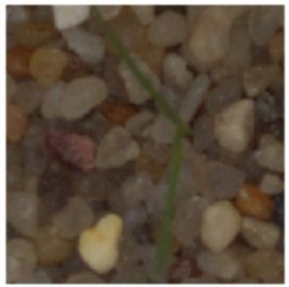
06 **Solution Novelty**

Innovative techniques implemented

Problem Statement

Given the RGB image of a plant seedling, try to identify it’s category i.e. species.

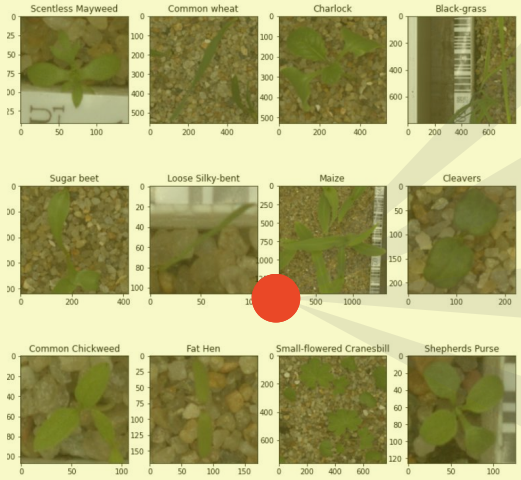





Exploratory Data Analysis

Overview

Image dataset



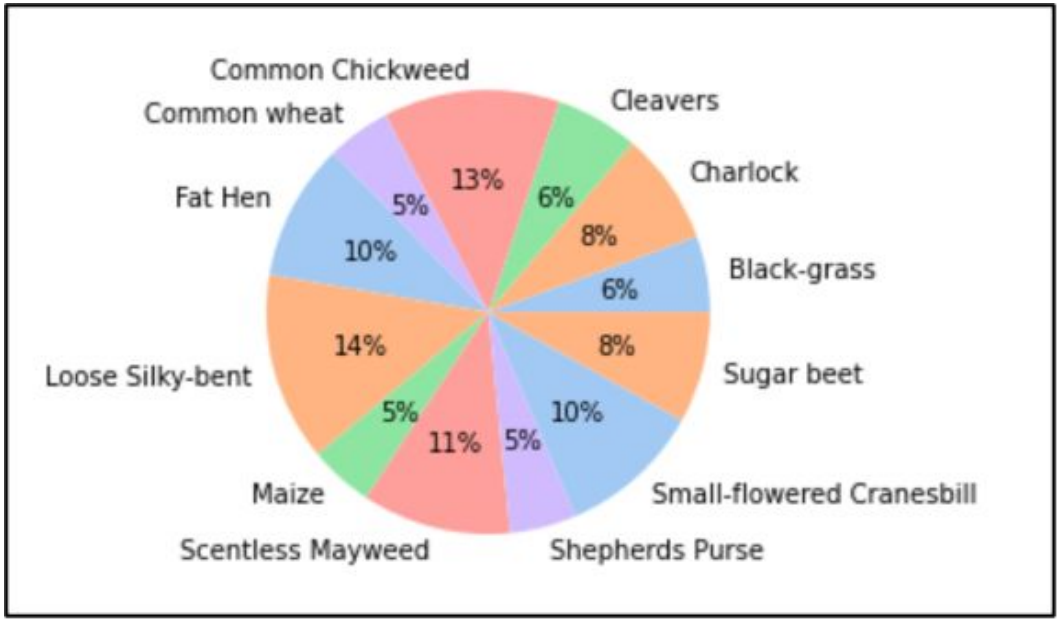
960 unique plants

12 species

5544 labelled images

Variable image sizes

Data Distribution



- Non-uniform distribution of images per species
- Almost 3 times as many images for one species in the training dataset as compared to another species
- Low number of training images
- May lead to model overfitting

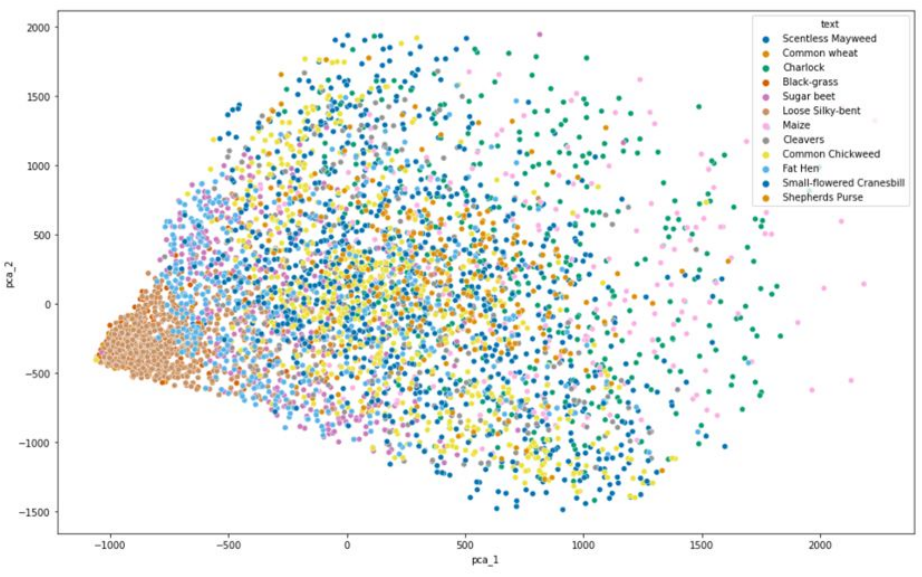
Principal Component Analysis

- Dimensionality-reduction technique
- Images have 45 x 45 x 3 = 6075 dimensions initially
- Try to identify the 2 principal components i.e. most important dimensions
- First component has the largest variance, second has a high variance along with low correlation with the first component

Principal Component Analysis (contd.)

- PCA components have a higher variance when applied on segmented images
- Models tested against segmented images as well to see if classification accuracy improves

Distribution of first 2 PCA components on original images



Distribution of first 2 PCA components on segmented images

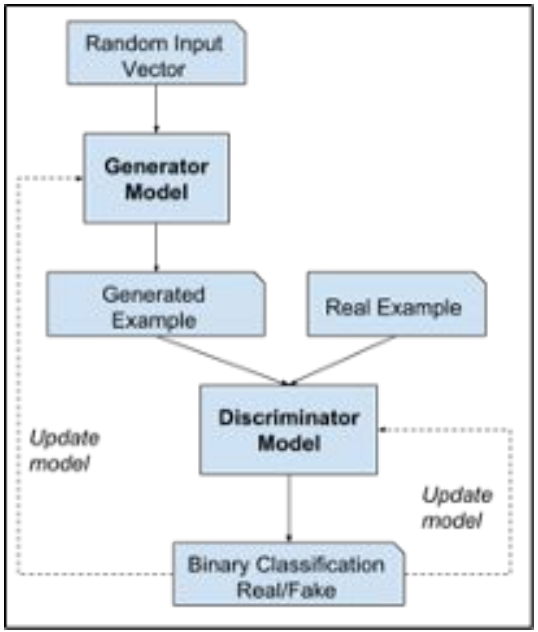
Data Preprocessing

Normalization & Resizing

- Pixel values normalized to a [0,1] range
  - Helps in faster convergence of gradient descent algorithms
  - Helps make features contribute equally in distance-based algorithms

- Images resized to specific sizes as required by the models being used
  - Machine learning models expect a fixed-size training dataset of images
  - Most images resized to 299 x 299 pixels, unless different sizing specified.

Balancing the dataset - Generative Adversarial Networks (GAN)

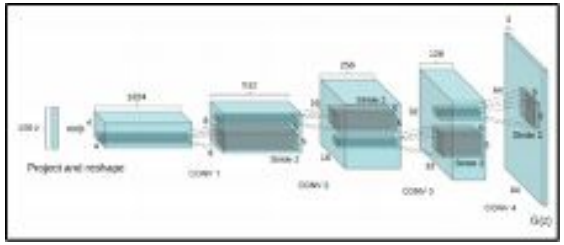


- Enhanced small image datasets by producing artificial data samples
- Consists of 2 distinct models- **Generator** and **Discriminator**
  - Generator produces fake images capturing properties of real images
  - Discriminator identifies if generated images are real or fake
- Both models keep improving during training
- Equilibrium is achieved when discriminator can no longer give a correct classification for the images outputted by the generator

Balancing the dataset - Generative Adversarial Networks (GAN)

Deep Convolutional GAN

- Uses convolutional and convolutional-transpose layers in the discriminator and generator respectively
- Generates images from random classes, which was a problem because we want to target particular species with less data
- Even though DCGAN generates real images, we can not use it to balance the dataset because images are generated from random classes, which may worsen the data imbalance



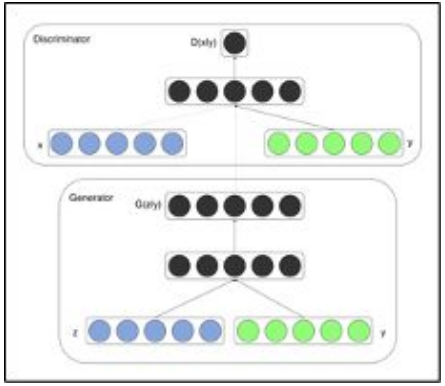
DCGAN Architecture

Example of real and fake images generated by DCGAN

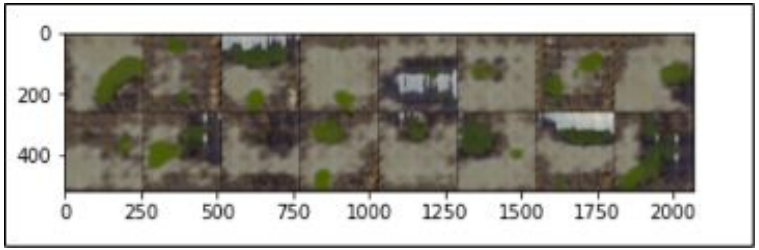
Balancing the dataset - Generative Adversarial Networks (GAN)

Conditional GAN

- Makes use of class labels in addition to the images as input both to the generator and discriminator models
- However, outputs of this model aren't desirable- the generator is unable to capture many axes of variation
- Most likely due to insufficient training instances for each class along with a skew in training data

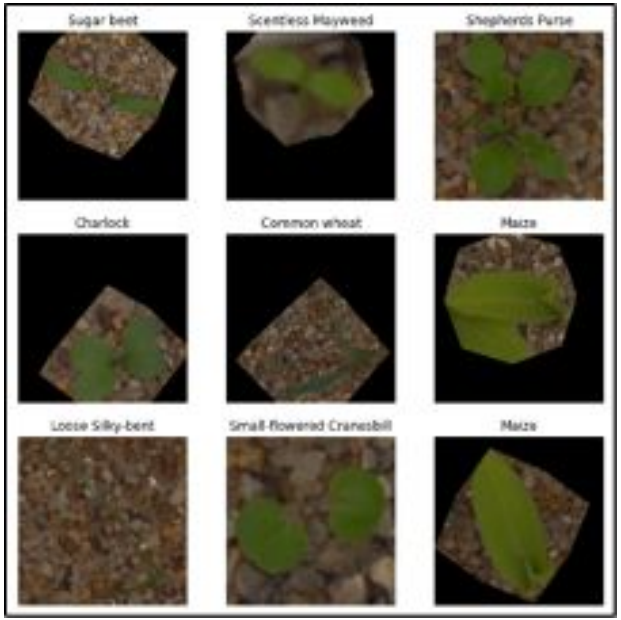


CGAN Architecture



Example of real and fake images generated by CGAN

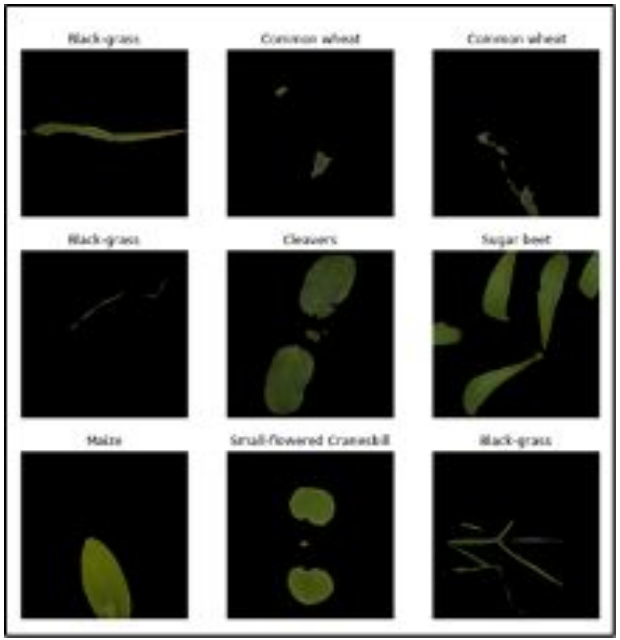
Balancing the dataset - Image Augmentation



Examples of augmented images

- Enhanced the training dataset by creating transformed versions of existing images
- Improves the performance and ability of the models to generalize
- Augmentation techniques
  - Random perspective transformation of the given image with a given probability.
  - Random rotation
  - Random affine transformation of the image keeping center invariant.
  - Image flipping (vertical and horizontal)

Balancing the dataset - Image Segmentation



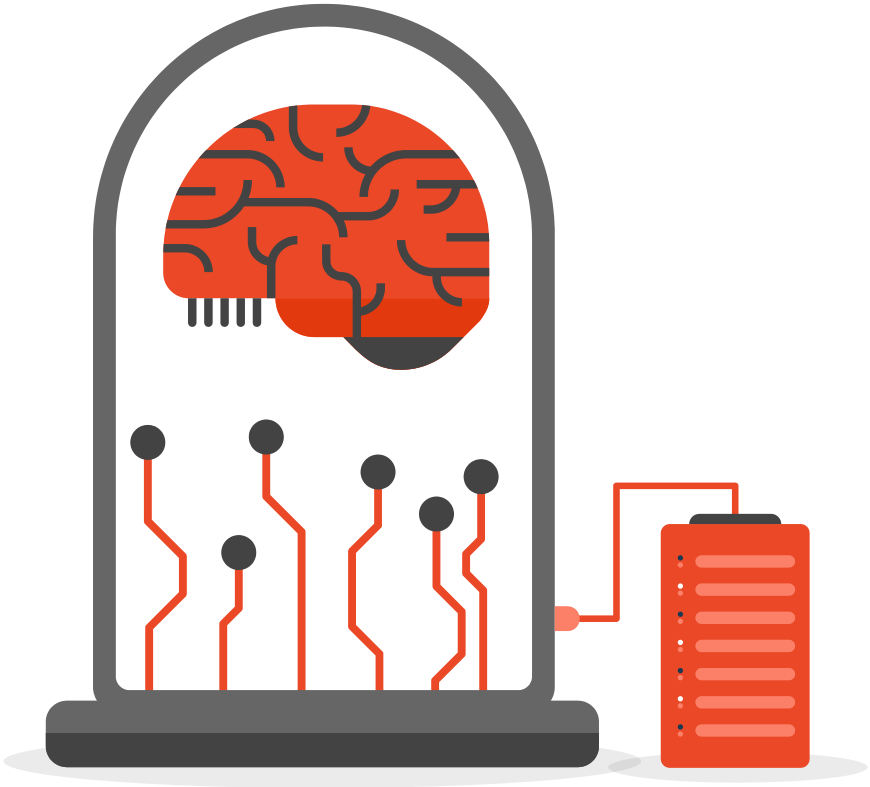
Examples of segmented images

- Used to partition images into different regions/segments under their respective class labels
- Highlights the segments better for the model to pick up
- Segmentation technique
  - We separated the background pixels from the foreground pixels, thus removing the noise from the images
  - Done using the Hue-Saturation-Value to create masks for plants to partition them
  - We also sharpened the segmentation mask to get better quality of images

Final preprocessed data

Four types of datasets prepared for training:

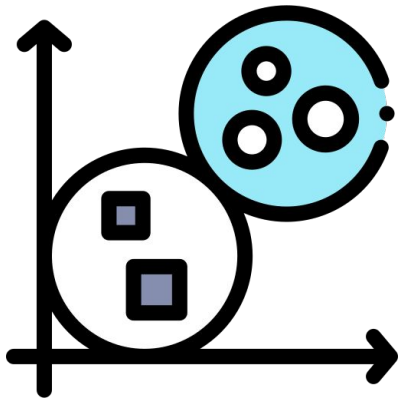
- Original dataset: provided on Kaggle
- Segmented dataset: original dataset with segmentation
- Balanced dataset: original dataset with augmentation
- Balanced segmented dataset: original dataset with both segmentation and augmentation (1000 images per class)



Solutions Implemented

Approach 1: Clustering Models

K-means, K-Nearest Neighbours



1.1 K-Means

Best Train Accuracy

0.3099

- An unsupervised learning algorithm for clustering data, based on data patterns
- Initial number of clusters set to K=12, so we will have 12 cluster centers
- Best performance on Segmented Dataset.

1.2 K-Nearest Neighbours (KNN)

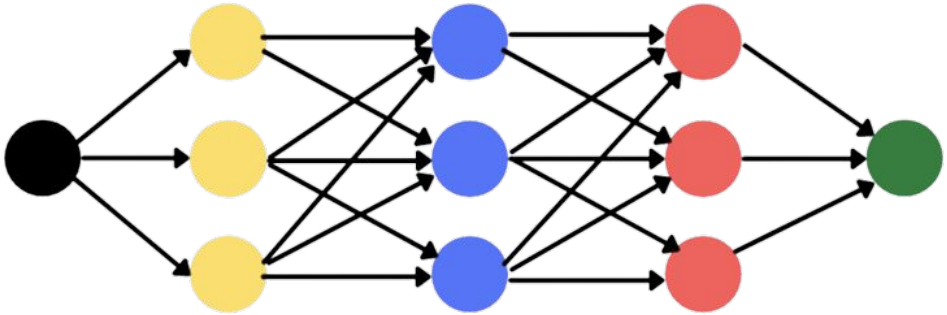
Best Public and Private Score

0.51196

- A supervised learning algorithm, that assumes that similar training instances exist in close proximity
- Used OneVsRestClassifier + KNeighboursClassifier to tackle the multiclass problem
- Best performance on Segmented Dataset.

Approach 2: CNN Model

Custom Convolutional Neural Network

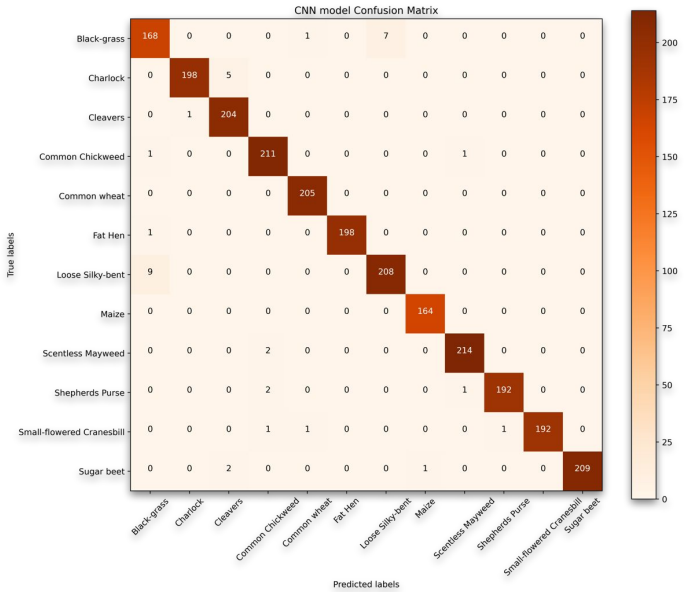


2.1 Custom Convolutional Neural Network Model

Best Public and Private Score

0.96725

Confusion Matrix

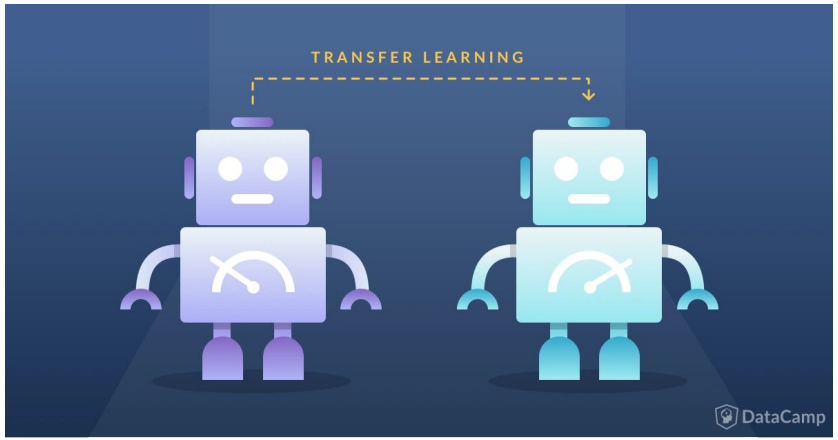


- CNN's specialize in processing data that has a grid-like topology, such as an image
- Basic CNN block:
  - Convolutional layer > Batch Normalization > Leaky Relu
- Key highlights:
  - Leaky Relu solves the problem of "dying ReLu"
  - Softmax activation is used for the last Dense layer
  - Tweaked beta1, beta2, epsilon for Adam
- Best performance on Segmented Dataset.

Image size	Epoch	Batch size	Optimizer	Loss function
128x128	175	16	Adam (lr=0.0002)	categorical_crossentropy

Approach 3: Transfer learning

EfficientNet, Xception, and Inception-ResNet-v2



Transfer learning overview

- *As stated in the Handbook of Research on Machine Learning Applications,* ***transfer learning*** *is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.*

- 3 different pre-trained models - trained on 4 datasets:

  EfficientNet, **Inception-Resnet-v2** and **Xception**

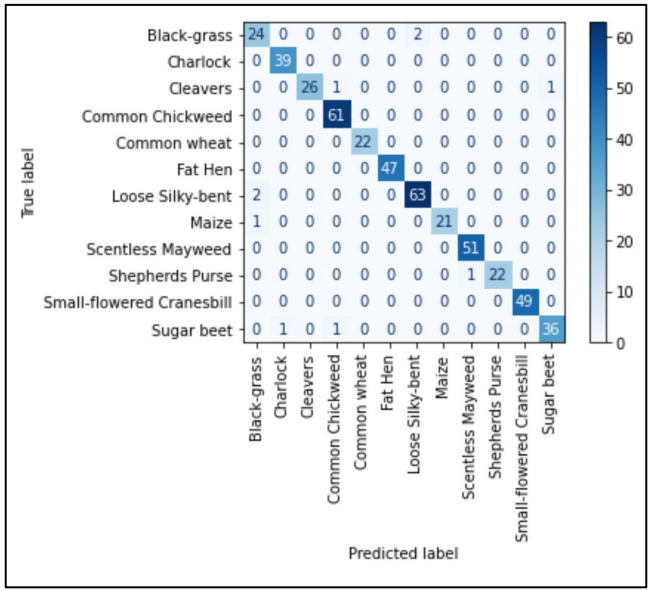
- General practices:
  - Unfreeze all the layers and train them along with the classification head
  - Some layers following the original model architectures:
    - Batch Normalization
    - Dropout.
  - Callback functions:
    - EarlyStopping
    - ReduceLROnPlateau

3.1 EfficientNet

Best Public and Private Score

0.98236

Confusion Matrix



- EfficientNets achieve state-of-the-art accuracy on ImageNet with an order of magnitude better efficiency
- Chosen version - EfficientNetB3 model
- Best performance on Original Dataset.

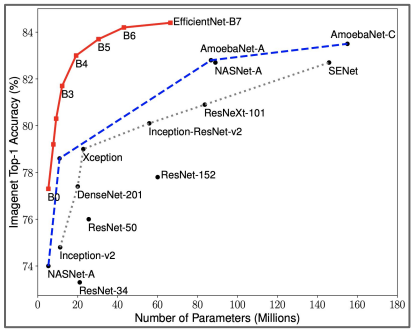


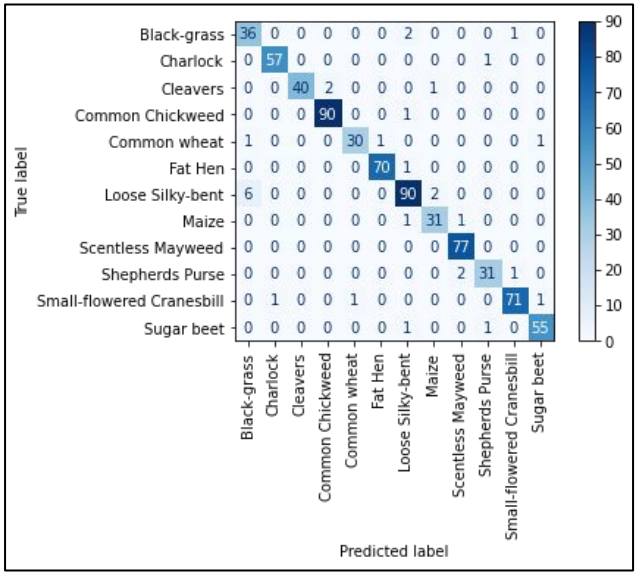
Image size	Epoch	Batch size	Optimizer	Loss function
299x299	40	16	Adam (lr=0.0001)	categorical_crossentropy

3.2 Inception-ResNet-v2

Best Public and Private Score

0.98488

Confusion Matrix



- Builds on the Inception family of architectures but incorporates residual connections
- Pre-trained with Imagenet dataset.
- Chosen version - Inception-ResNet-v2 model
- Use Mish activation function:
  - Prevent saturation due to capping
  - Better gradient flow
- Best performance on Balanced Dataset

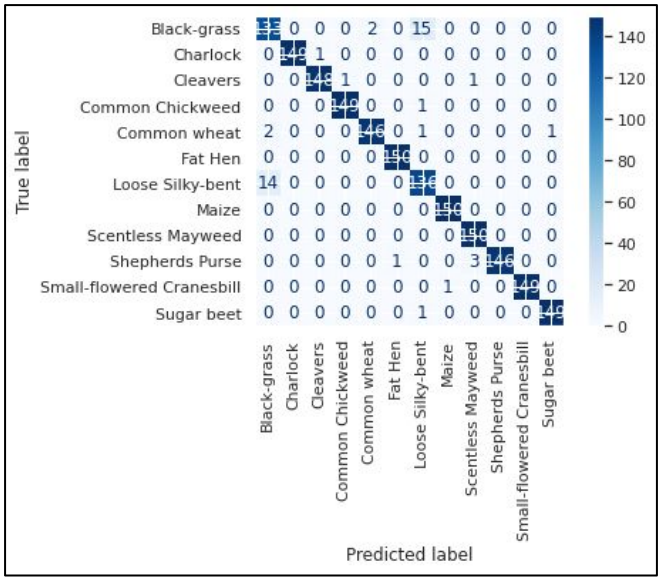
Image size	Epoch	Batch size	Optimizer	Loss function
299x299	100	16	Adam (lr=0.0001)	categorical_crossentropy

3.3 Xception

Best Public and Private Score

0.97858

Confusion Matrix



- Transfer learning on pretrained version of the network on ImageNet database
- Use Scaled Exponential Linear Units (SELU) activation function:
  - converge faster
  - prevent vanishing and exploding gradient problem.
- Best performance on Balanced Dataset Segmented

Image size	Epoch	Batch size	Optimizer	Loss function
299x299	40	16	Adam (lr=0.0001)	categorical_crossentropy

Approach 4: Vision Transformers

Custom Vision Transformer, Pre-trained transformer

Transformers

4.1 Vision Transformer

Best Public and Private Score

0.87027

Hyper-parameters	Value
Embedding dimensions	256
Hidden Dimensions	512
Number of heads	8
Number of layers	6
Patch size	16x16
Number of channels	3
Number of patches	196
Dropout Rate	25%

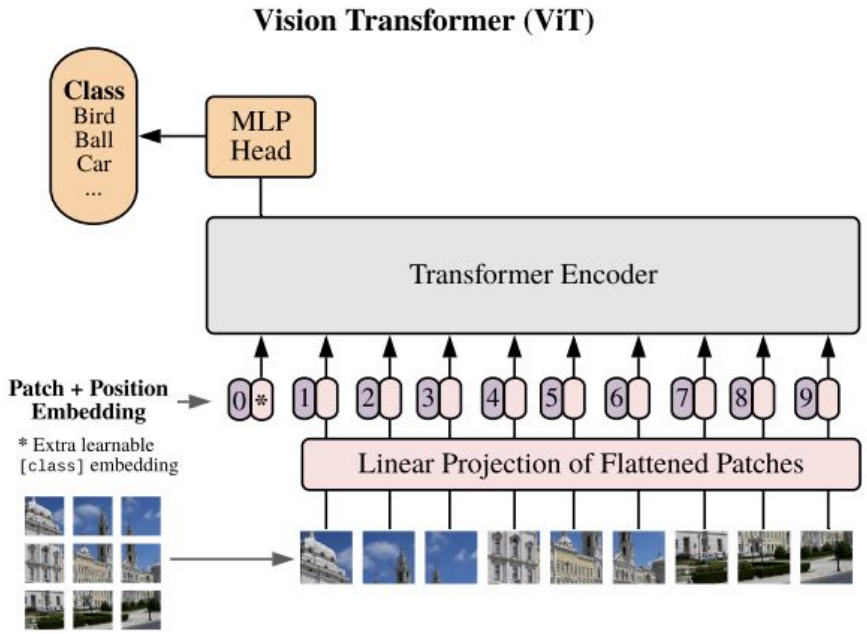
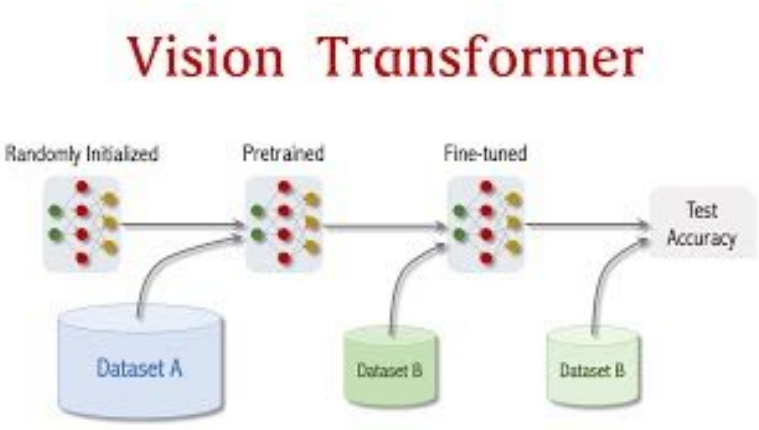


Image size	Epoch	Batch size	Optimizer	Loss function
224x224	60	16	Adam (lr=0.0003)	Categorical Crossentropy

4.2 Pre-trained Vision Transformer

Best Public and Private Score

0.98110

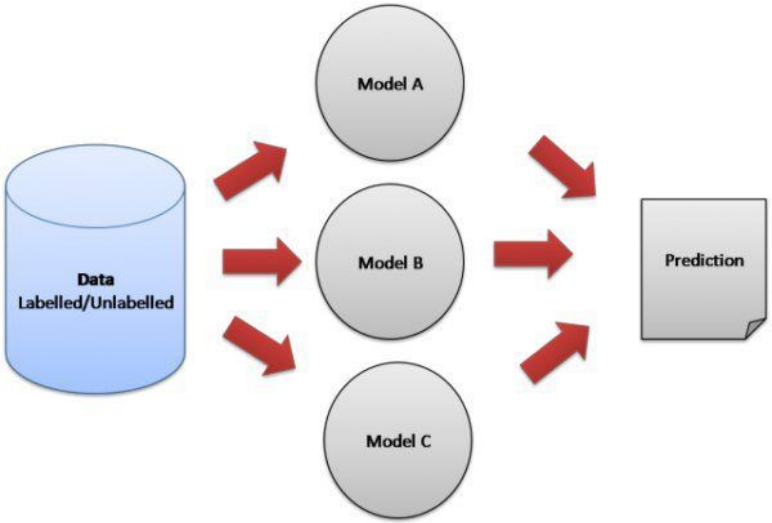


- Vision Transformers achieve high performances only when trained on large dataset
- Used HuggingFace pre-trained vision transformer trained on ImageNet-21k(12 million images, 21843 classes)
- Best performance on Balanced Dataset.

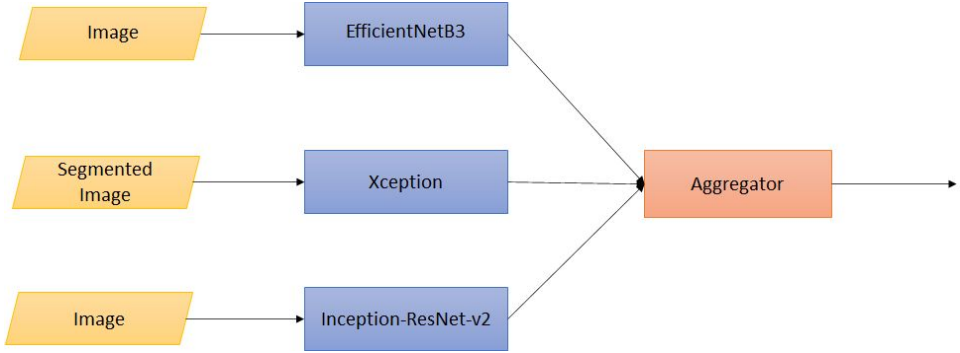
Image size	Epoch	Batch size	Optimizer	Loss function
224x224	100	16	Adam (lr=0.00005)	Categorical Crossentropy

Approach 5: Ensembling

Weighted Average Ensemble



Weighted Average Ensemble



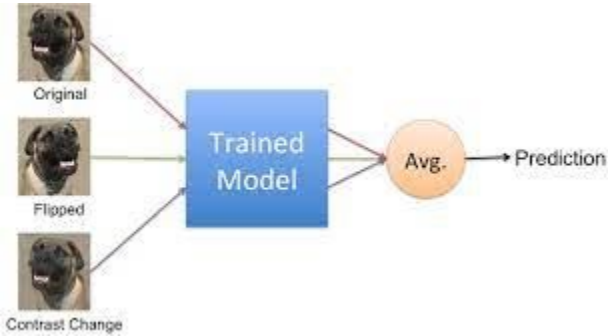
Efficient NetB3	Inception -ResNet- v2	Xception	Public score	Private Score
0.33	0.33	0.33	0.98614	0.98614
0.25	0.5	0.25	0.98992	0.98992
0.1	**0.5**	**0.4**	**0.99118**	**0.99118**
0.3	0.5	0.2	0.98992	0.98992
0.3	0.6	0.1	0.98740	0.98740
0.2	0.6	0.2	0.98866	0.98866

$$P(x) = 0.1 \cdot EfficientNetB3(x) + 0.5 \cdot InceptionResNetV2(x) + 0.4 \cdot Xception(x)$$

Other Techniques

Test-Time Augmentation, Binary Classifier Error Correction.

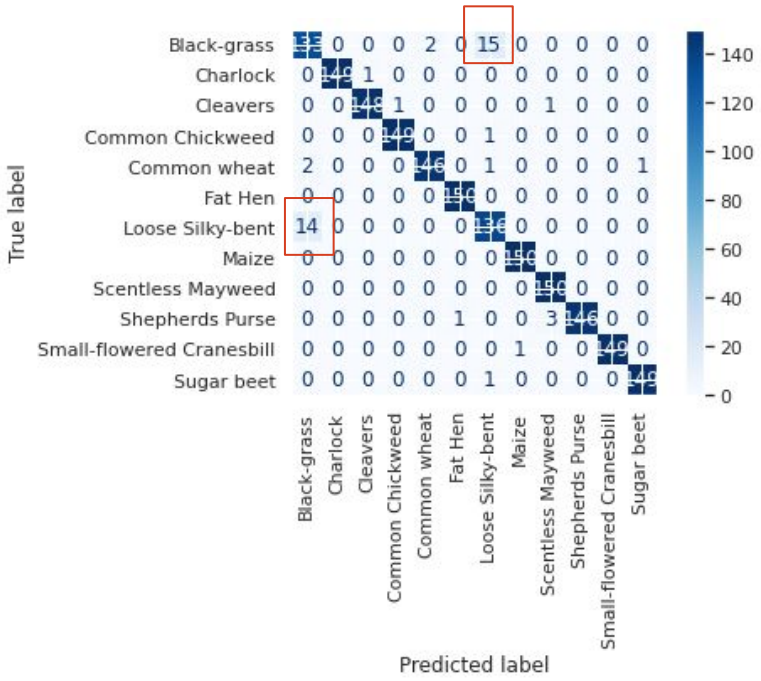
Test-Time Augmentation



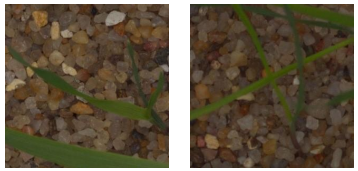
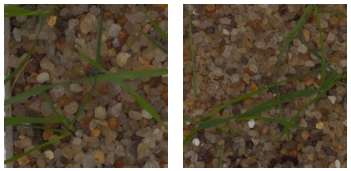
- 20 Augmentations of test image
- Implemented with keras datagenerator and pytorch dataloader
- Reached Top 4.8% (Tank 40/833) with Inception-ResNet-v2 + TTA

Model	Public Score	Private Score	Public Score with TTA	Private score with TTA
Convolutional Neural Network	0.96725	0.96725	**0.96851**	**0.96851**
Xception	0.97481	0.97481	**0.98362**	**0.98362**
Inception-Res net-v2	0.98488	0.98488	**0.98614**	**0.98614**
EfficientnetB3	**0.98362**	**0.98362**	0.97984	0.97984
ViT	**0.98110**	**0.98110**	0.97355	0.97355

Misclassification

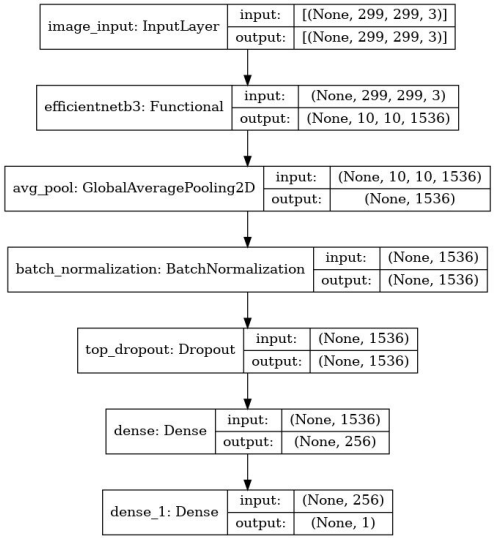


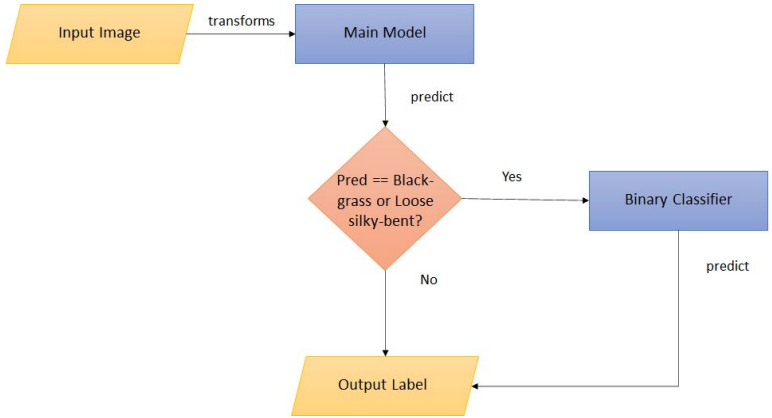
- Black-grass and Loose-silky bent are often misclassified as the other with all our models
- Images of Black-grass and Loose Silky bent look identical

Black-grass	Loose Silky-bent
	

Binary Classifier

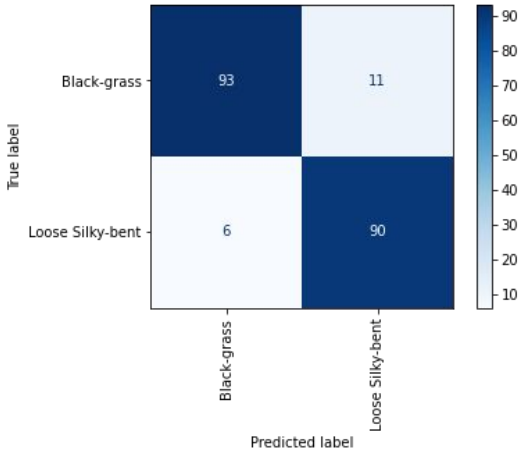
- Binary Classifier trained on Black-grass and Loose Silky-bent
- Modifies predictions of main model





- EfficientNetB3 backbone with Batch Normalization and Dropouts followed by a classification layer with 1 neuron and sigmoid activation function

Results



Binary Classifier mis-classifies Black-grass and Loose-Silky bent

Binary Classifier worsened the performance of all our models

Model	Public Score	Private Score	Public Score with Binary Classifier	Private score with Binary Classifier
Convolutional Neural Network	**0.96725**	**0.96725**	0.85012	0.85012
Xception	**0.97858**	**0.97858**	0.96725	0.96725
Inception-Res net-v2	**0.98488**	**0.98488**	0.97607	0.97607
EfficientnetB3	**0.98362**	**0.98362**	0.97355	0.97355

Leaderboard

Top 1.56% (13/833)

YOUR RECENT SUBMISSION

efficientnetb3_inceptionresnetv2_xception_0.1_0.5_0.4_submission.csv

Score: 0.99118

Submitted by Atul1706 · Submitted just now

↓ Jump to your leaderboard position

Solution Novelty

01

Enhanced Activation Function

02

Dropouts, BatchNorm

03

Callbacks - EarlyStopping, ReduceLROnPlateau

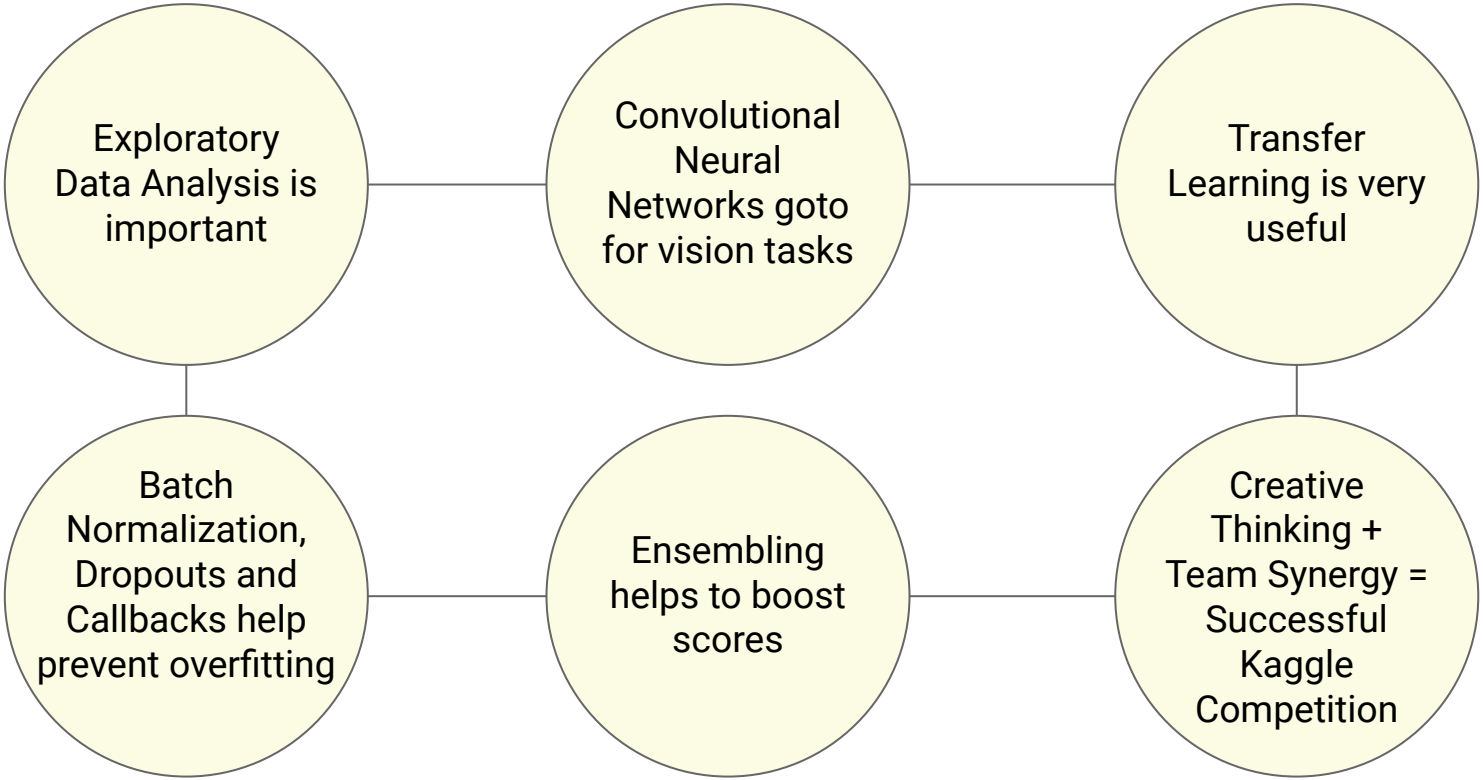
04

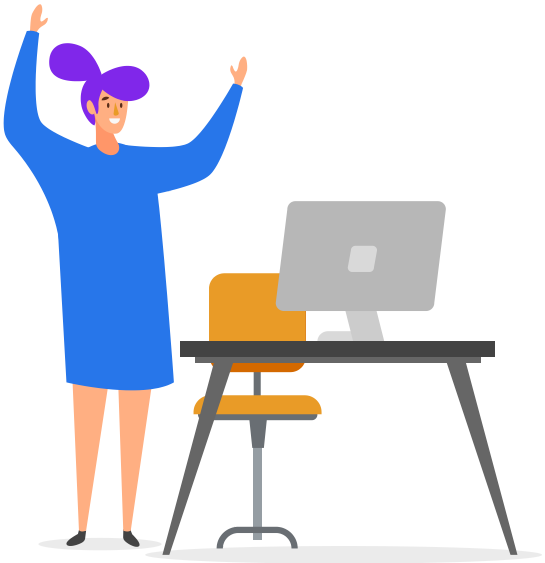
Ensemble

05

Transformers

Conclusion


Exploratory Data Analysis is important
Convolutional Neural Networks goto for vision tasks
Transfer Learning is very useful
Batch Normalization, Dropouts and Callbacks help prevent overfitting
Ensembling helps to boost scores
Creative Thinking + Team Synergy = Successful Kaggle Competition



Thank you!