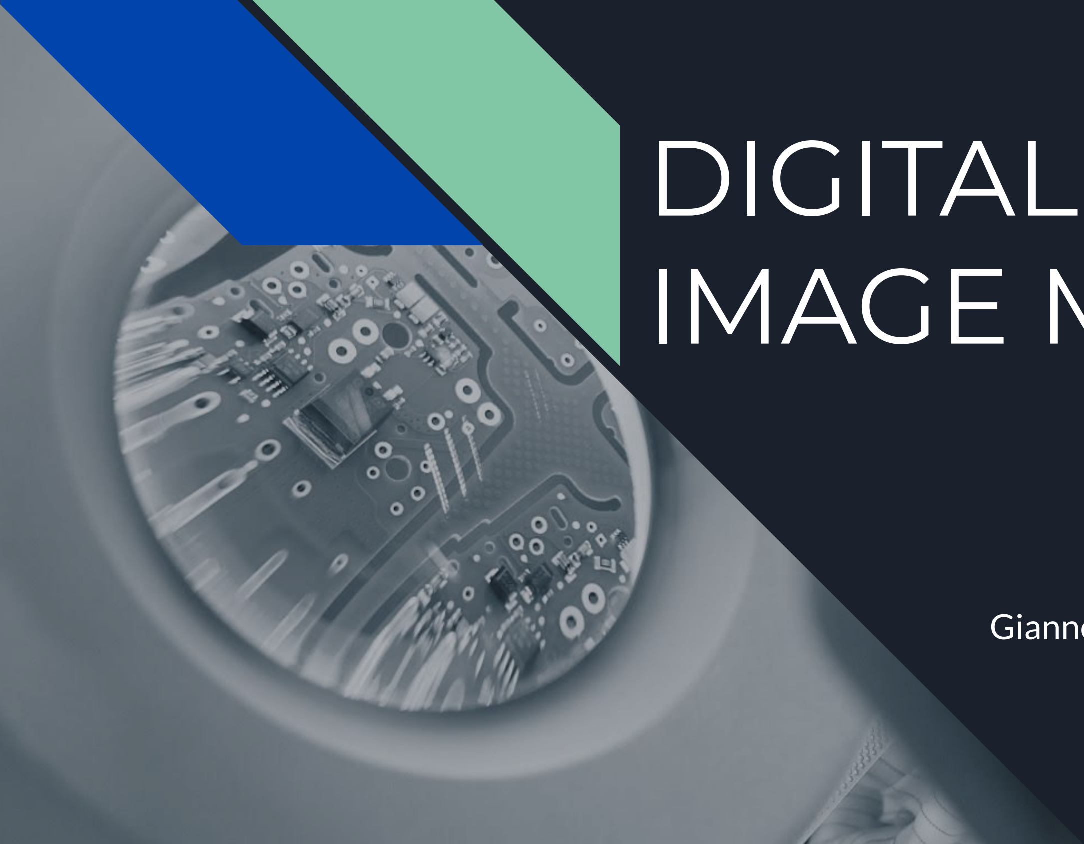


DIGITAL SIGNAL & IMAGE MANAGEMENT

Giannelli Alessio   Imbonati Lorenzo   Valoti Davide

SUMMARY

TASK 1: Audio Signal Classification - Speech Commands Dataset

TASK 2: Image Classification - FGVC Aircraft 100 Dataset using Densenet201

TASK 3: Image Retrieval - Feature Re-Weighting in CBIR

TASK 1 - Dataset & Pre-Processing

Info about dataset

Dataset size ≈ 2GB

N. of istances = 64720

N. of istances per class ≈ 2150

30 Class Labels

```
[array(['bed', 'bird', 'cat', 'dog', 'down', 'eight', 'five', 'four', 'go'
       'happy', 'house', 'left', 'marvin', 'nine', 'no', 'off', 'on',
       'one', 'right', 'seven', 'sheila', 'six', 'stop', 'three', 'tree',
       'two', 'up', 'wow', 'yes', 'zero'], dtype='<U6')]
```

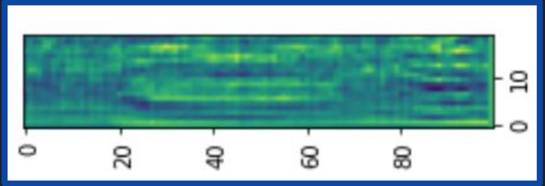
Splitting

TRAIN (80%)

TEST (20%)

Features Extraction

```
def feats_mfcc(input, sample_rate):
  cepstral_features = mfcc(input, sample_rate, numcep=20) #Features extraction
  zero_vector = np.zeros((100-cepstral_features.shape[0], cepstral_features.shape[1]))
  cepstral_features = np.vstack((cepstral_features, zero_vector))
  return cepstral_features
```



Example of MFCC Feature

TASK 1 - Model Architecture

Layers and Parameters

```
net.summary()

Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 100, 20)]         0

 gru_2 (GRU)                 (None, 50)                10800

 dense_2 (Dense)             (None, 30)                1530

=================================================================
Total params: 12,330
Trainable params: 12,330
Non-trainable params: 0
_________________________________________________________________
```

Dropout = 0.3

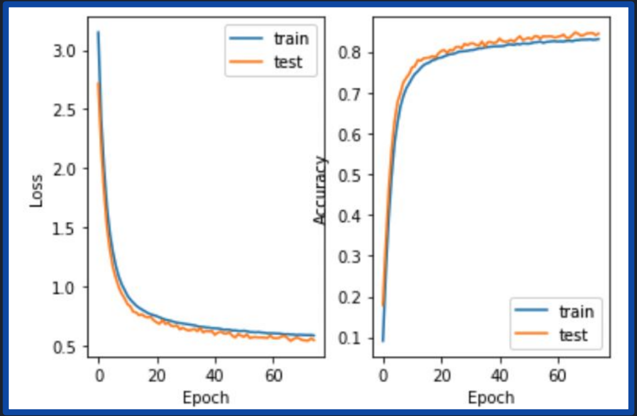
Activation = Softmax

TASK 1 - Model Performance

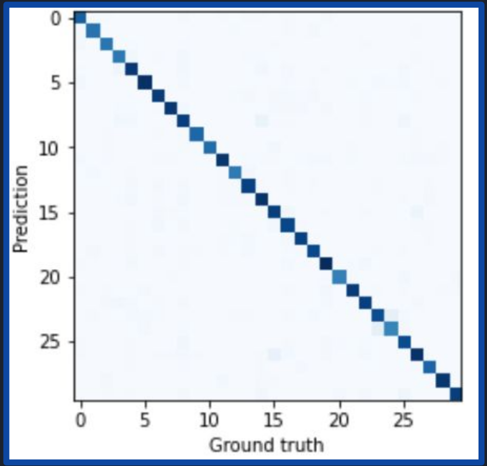
Classification Report

	precision	recall	f1-score	support
0.0	0.89	0.92	0.90	391
1.0	0.94	0.92	0.93	351
2.0	0.92	0.94	0.93	341
3.0	0.88	0.91	0.90	345
4.0	0.86	0.90	0.88	484
5.0	0.95	0.92	0.93	492
accuracy			0.91	12943
macro avg	0.92	0.91	0.91	12943
weighted avg	0.91	0.91	0.91	12943

Loss & Accuracy Trend



Confusion Matrix



Very high performance, touching 90% accuracy on both training and validation set.

TASK 1 - Model Evaluation

Audio Prediction

```
np.argmax(previsione)

10
```

```
encoder.inverse_transform([[10]])

array([['house']], dtype='<U6')
```

Distribution Probability

```
previsione

array([[3.9753129e-05, 5.0238521e-07, 1.1391650e-06, 2.7455920e-03,
        4.6533391e-06, 3.7695609e-02, 1.2968192e-07, 2.0860985e-05,
        2.8214217e-04, 8.0605365e-05, 4.8249230e-01, 3.7027390e-03,
        2.0234948e-08, 7.9701730e-04, 1.9732230e-04, 3.9731449e-01,
        1.7278563e-02, 1.7386535e-02, 4.9062535e-05, 1.8000244e-05,
        3.3243868e-02, 7.7199639e-04, 3.0823336e-03, 2.3951443e-04,
        6.7771517e-04, 1.0602005e-04, 5.4005993e-04, 7.4405796e-07,
        1.1608218e-03, 6.9894508e-05]], dtype=float32)
```

The model correctly predicts audio with the "house" class
with a probability of almost 50%.

TASK 2 - Dataset & Pre-Processing

Info about Dataset

Dataset Size: 2.76 GB

Number of classes: 100

Number of Instances per class: 100

Cropping

Splitting

TRAIN (60%) 6000 instances	VALIDATION (30%) 3000 instances	TEST (10%) 1000 instances

TASK 2 - Model ArchitectureV1

Transfer Architecture DenseNet201

```
base_net = keras.applications.DenseNet201(input_shape=(224,224,3),weights='imagenet', include_top=False)
for layer in base_net.layers[:501]:
  layer.trainable = False
```

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 224, 224, 3)]	0
tf.math.truediv_4 (TFOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add_2 (TFOpLambda)	(None, 224, 224, 3)	0
tf.math.truediv_5 (TFOpLambda)	(None, 224, 224, 3)	0
densenet201 (Functional)	(None, 7, 7, 1920)	18321984
average_pooling2d_2 (AveragePooling2D)	(None, 3, 3, 1920)	0
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1920)	0
dense_6 (Dense)	(None, 1024)	1967104
batch_normalization_4 (BatchNormalization)	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524800
batch_normalization_5 (BatchNormalization)	(None, 512)	2048
dropout_5 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 100)	51300

Total params: 20,871,332
Trainable params: 9,055,396
Non-trainable params: 11,815,936

Transfer Learning

TASK 2 - Model ArchitectureV2

Transfer Architecture DenseNet201

```
base_net = keras.applications.DenseNet201(input_shape=(224,224,3),weights='imagenet', include_top=False)
base_net.trainable = True
```

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 224, 224, 3)]	0
tf.math.truediv_2 (TFOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add_1 (TFOpLambda)	(None, 224, 224, 3)	0
tf.math.truediv_3 (TFOpLambda)	(None, 224, 224, 3)	0
densenet201 (Functional)	(None, 7, 7, 1920)	18321984
average_pooling2d_1 (AveragePooling2D)	(None, 3, 3, 1920)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1920)	0
dense_3 (Dense)	(None, 1024)	1967104
batch_normalization_2 (BatchNormalization)	(None, 1024)	4096
dropout_2 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 512)	524800
batch_normalization_3 (BatchNormalization)	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 100)	51300

```
Total params: 20,871,332
Trainable params: 20,639,204
Non-trainable params: 232,128
```

Transfer Learning

TASK 2 - Model ArchitectureV3

Transfer Architecture ResNet50

```
base_net = keras.applications.ResNet50(input_shape=(224,224,3),weights='imagenet', include_top=False)
base_net.trainable = True
```

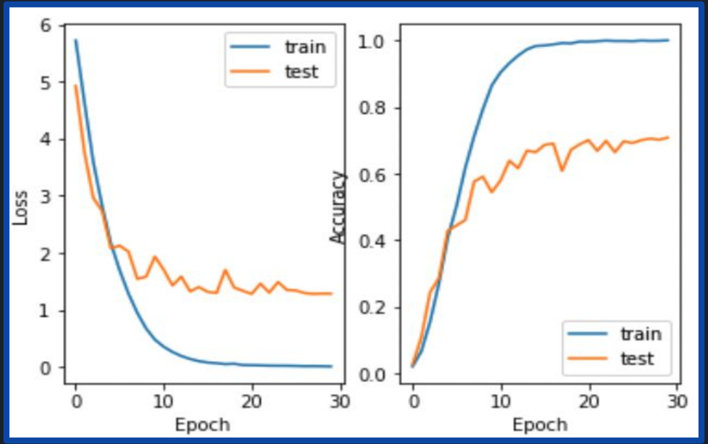
```
Layer (type)                 Output Shape              Param #
===============================================================
input_9 (InputLayer)         [(None, 224, 224, 3)]     0

tf.__operators__.getitem_1    (None, 224, 224, 3)      0
(SlicingOpLambda)

tf.nn.bias_add_4 (TFOpLambd   (None, 224, 224, 3)      0
a)

resnet50 (Functional)        (None, 7, 7, 2048)        23587712

average_pooling2d_4 (Averag   (None, 3, 3, 2048)       0
ePooling2D)

global_average_pooling2d_4    (None, 2048)             0
(GlobalAveragePooling2D)

dense_12 (Dense)             (None, 1024)              2098176

batch_normalization_8 (Batc   (None, 1024)             4096
hNormalization)

dropout_8 (Dropout)          (None, 1024)              0

dense_13 (Dense)             (None, 512)               524800

batch_normalization_9 (Batc   (None, 512)              2048
hNormalization)

dropout_9 (Dropout)          (None, 512)               0

dense_14 (Dense)             (None, 100)               51300
```

```
Total params: 26,268,132
Trainable params: 26,211,940
Non-trainable params: 56,192
```

Transfer Learning

TASK 2 - Model Performance V1 DenseNet201 9M trainable parameters

Loss & Accuracy Trend



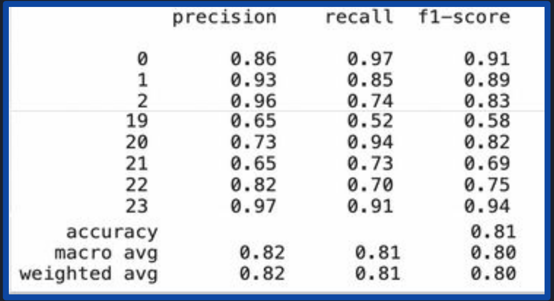
Test loss	Test accuracy
1.289	70.75%

The performance of the model is fair since it achieves almost 100% on the training set against 65-70% accuracy on the validation set.

It is emphasized, however, that from the 10th epoch onward there is a risk of overfitting problem.

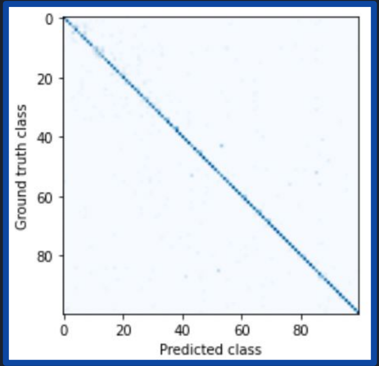
TASK 2 - Model Performance V2
DenseNet201 20M trainable parameters

Classification Report

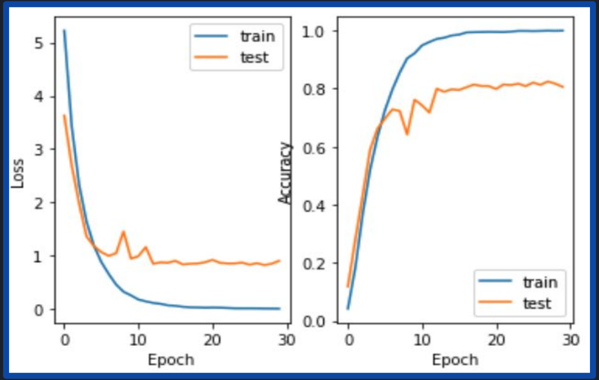


	precision	recall	f1-score
0	0.86	0.97	0.91
1	0.93	0.85	0.89
2	0.96	0.74	0.83
19	0.65	0.52	0.58
20	0.73	0.94	0.82
21	0.65	0.73	0.69
22	0.82	0.70	0.75
23	0.97	0.91	0.94
accuracy			0.81
macro avg	0.82	0.81	0.80
weighted avg	0.82	0.81	0.80

Confusion Matrix



Loss & Accuracy Trend



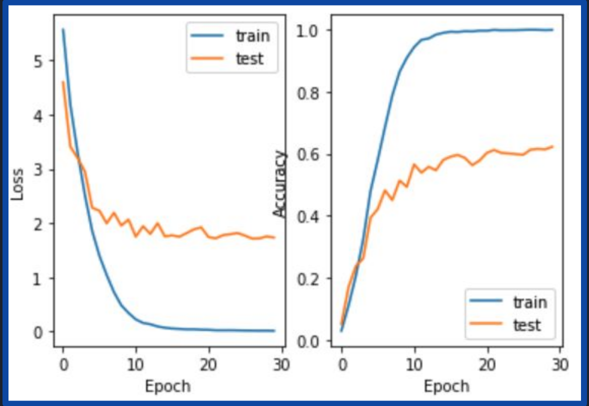
The performance of the model is very good as it achieves 80% accuracy on the validation set.
The trend shows some performance fluctuations but in general it is quite stable and the validation follows the growth of the training set.

TASK 2 - Model Performance V3
ResNet50 26M trainable parameters

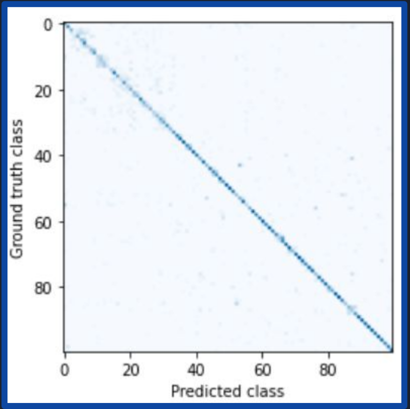
Classification Report

	precision	recall	f1-score
0	0.34	0.67	0.45
1	0.67	0.67	0.67
2	0.56	0.44	0.49
97	0.92	0.70	0.79
98	0.74	0.74	0.74
99	0.82	0.82	0.82
accuracy			0.61
macro avg	0.62	0.61	0.61
weighted avg	0.62	0.61	0.61

Loss & Accuracy Trend



Confusion Matrix



The performance of the model is not acceptable since it achieves almost 100% accuracy on the training set but settles around 60% on the validation set.

In general, it seems that the model suffers from overfitting.

TASK 2 - Model Evaluation

on Test Set

Test loss	Test accuracy
0.8971	80.30%

on Web Image

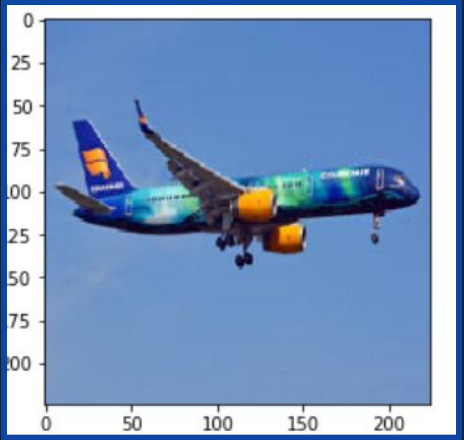
Distribution Probability

```
previsione

array([[8.45307895e-08, 8.59294147e-09, 2.28450489e-07, 1.04331457e-05,
        5.10848849e-07, 9.47635385e-07, 9.56761141e-08, 1.16615745e-04,
        1.19382069e-01, 6.69421115e-06, 1.71462332e-07, 1.86927124e-07,
        3.83096435e-08, 1.61001561e-04, 8.33525717e-01, 1.18540612e-03,
```

Image Prevision

```
np.argmax(previsione)

14
```

```
train.class_names[14]

'757-200'
```



TASK 3 - Feature Re-Weighting in CBIR

[Implementation of the following paper]

Feature Re-weighting in Content-Based Image Retrieval

Gita Das[1], Sid Ray[1], and Campbell Wilson[2]

[1] Clayton School of Information Technology
Monash University
Victoria 3800, Australia
{Gita.Das, Sid.Ray}@csse.monash.edu.au
[2] Caulfield School of Information Technology
Monash University
Victoria 3800, Australia
Campbell.Wilson@csse.monash.edu.au

Main concepts:

- Use of the previous neural network as feature extractor
- Use of weighted Minkowski distance as similarity measure
- Update of the query results according to user preferences

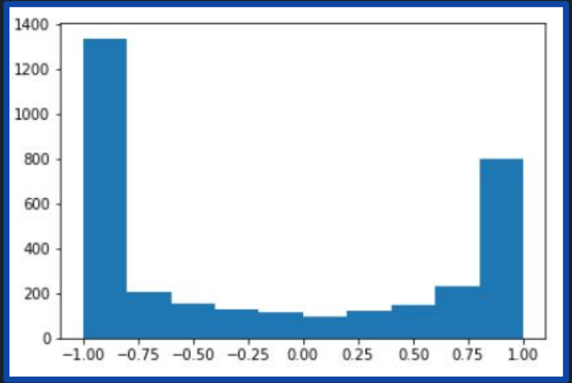
TASK 3 - Feature Extraction

Load Task2 Model

```
temp = keras.models.load_model('Model/densenet201_final_task2.h5')

layer_name = 'dense_1'
newmodel = Model(inputs=temp.input, outputs=temp.get_layer(layer_name).output)
newmodel.summary()
```

Splitting

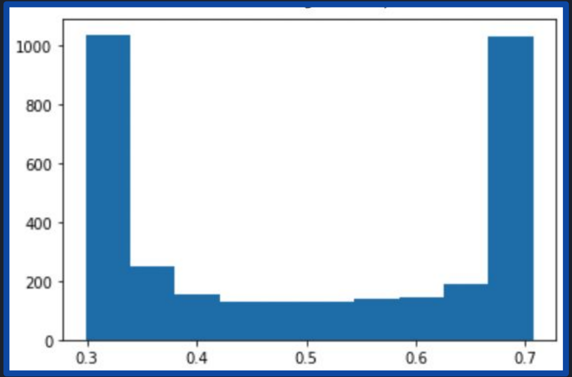
TRAIN	TEST
6000 instances	1000 instances



Features normalization

$$f_i' = \frac{f_{i,org} - \mu_i}{3\sigma_i}$$

$$f_i = \frac{f_i' + 1}{2}$$



TASK 3 - Image Retrieval (Query)

Manhattan similarity measure

$$D(I,Q) = \sum_{i=1}^{M} w_i * |f_{iI} - f_{iQ}|$$

Weights are constant for the first round of retrieval

Top20 Accuracy on test set

77,56%

Images Similarity to the test image

TASK 3 - Rebalancing type 1

Update weights formula Type 1

$$weight - type1 : w_i^{k+1} = \frac{\epsilon + \sigma_{N_r,i}^k}{\epsilon + \sigma_{rel,i}^k}, \epsilon = 0.0001$$

New weight for the i-th feature is equal to the division between the standard deviation over the 20 retrieved images and the standard deviation over the relevant images at the previous round

$$\mathbf{w^{k+1} = 0.9*w^k + 0.1*w^{k+1}}$$

Round number	Top20 Precision
Round 0	77.56
Round 1	83.94
Round 2	84.56
Round 3	85.10
Round 4	85.41
Round 5	**85.54**

TASK 3 - Rebalancing type 2

Update weights formula Type 2

$$weight - type2 : w_i^{k+1} = \frac{\delta_i^k}{\epsilon + \sigma_{rel,i}^k}$$

$$\delta_i^k = 1 - \frac{\sum_{l=1}^k |\psi_i^{l,U}|}{\sum_{l=1}^k |F_i^{l,U}|}$$

New weight for the i-th feature is equal to the division between the sigma quantity defined in the second formula, that depends on the **dominant range,** and the standard deviation over the relevant images at the previous round

$w^{k+1} = 0.9 * w^k + 0.1 * w^{k+1}$

Round number	Top20 Precision
Round 0	**77.56**
Round 1	61.70
Round 2	58.84
Round 3	59.91
Round 4	60.09
Round 5	60.53

TASK 3 - Rebalancing type 3

Update weights formula Type 3

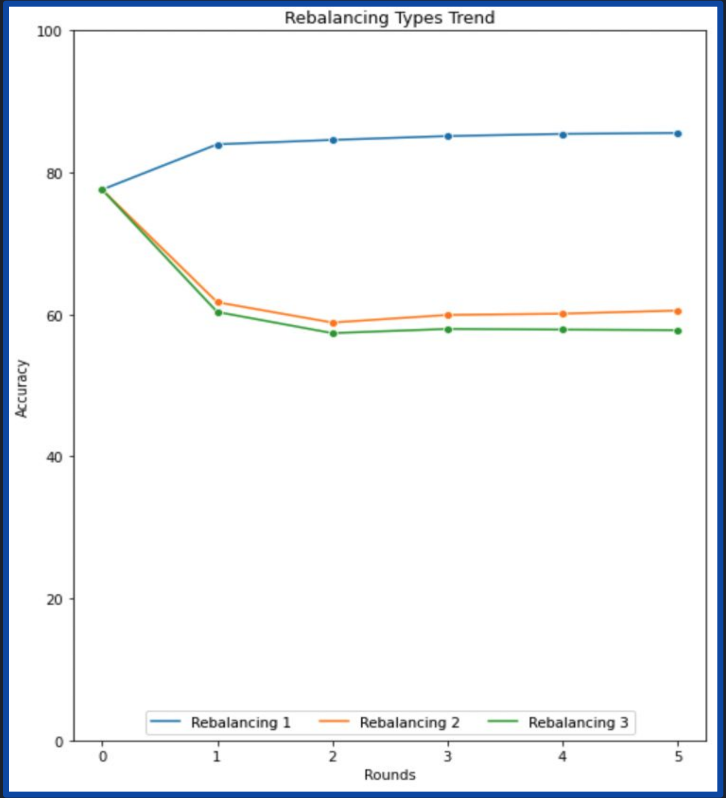
$$weight - type3 : w_i^{k+1} = \delta_i^k * \frac{\epsilon + \sigma_{N_r,i}^k}{\epsilon + \sigma_{rel,i}^k}$$

New weight for the i-th feature is equal to the the delta value defined in the previous slide by the weights of type 1

$$w^{k+1} = 0.9*w^k + 0.1*w^{k+1}$$

Round number	Top20 Precision
Round 0	**77.56**
Round 1	60.33
Round 2	57.35
Round 3	57.94
Round 4	57.85
Round 5	57.77

TASK 3 - Rebalancing Types Trend



Type 1 rebalancing is definitely the best since it shows increasing growth.

The other two types of rebalancing do not produce any improvement.

Let's leave room for the demo...