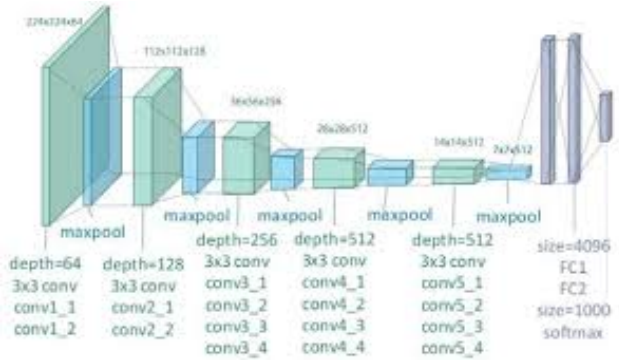
Cars type detection

- Dataset:
  - Link: https://www.kaggle.com/datasets/jutrera/stanford-car-dataset-by-classes-folder
  - We used only on the first 20 classes of total 819 training photo and applied augmentation on them so they reached 3276 training photo
- Models:
  - VGG-19:
    - Step-by-step explanation:
      - Input: Accepts a RGB image resized to 224x224 pixels.
      - Convolutional Blocks:
        - Block 1: 2 convolutional layers (3x3 filters, 64 channels) followed by max pooling (2x2).
        - Block 2: 2 convolutional layers (3x3 filters, 128 channels) followed by max pooling.
        - Block 3: 4 convolutional layers (3x3 filters, 256 channels) followed by max pooling.
        - Block 4: 4 convolutional layers (3x3 filters, 512 channels) followed by max pooling.
        - Block 5: 4 convolutional layers (3x3 filters, 512 channels) followed by max pooling.
        - Each convolution uses ReLU activation; pooling reduces spatial dimensions by half.
      - Fully Connected Layers:
        - Flatten the output from Block 5.
        - Layer 1: 4096 neurons, ReLU.
        - Layer 2: 4096 neurons, ReLU.
        - Layer 3: 20 neurons (for 20 classes), softmax for probabilities.
    - Architecture Diagram:



    - Reference: https://arxiv.org/abs/1409.1556

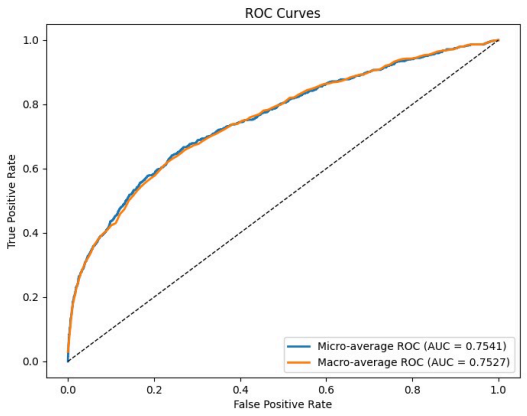
- Results after training from scratch:

```
Epoch [90/90] Train Loss: 0.6192 Train Acc: 0.9991 Test Acc: 0.2935
  → Checkpoint saved: checkpoints/model_epoch_90.pth

============================================================
Training Complete!
============================================================
Best test accuracy: 0.2947
Best model saved to: checkpoints/best_model.pth
```

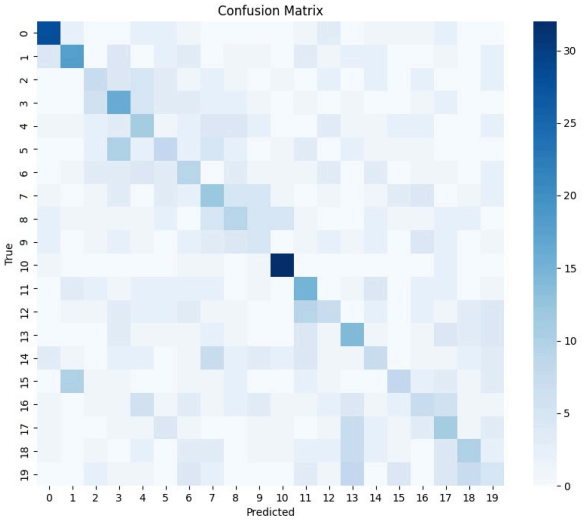
-

```
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.64      0.63        44
           1       0.47      0.41      0.44        44
           2       0.23      0.22      0.22        32
           3       0.27      0.37      0.31        43
           4       0.23      0.26      0.24        42
           5       0.22      0.20      0.21        40
           6       0.20      0.23      0.21        39
           7       0.23      0.27      0.24        45
           8       0.23      0.22      0.23        41
           9       0.17      0.15      0.16        33
          10       0.70      0.84      0.76        38
          11       0.29      0.38      0.33        40
          12       0.23      0.17      0.19        42
          13       0.29      0.34      0.31        41
          14       0.21      0.16      0.18        43
          15       0.33      0.22      0.27        36
          16       0.20      0.16      0.18        45
          17       0.23      0.28      0.25        39
          18       0.32      0.24      0.27        42
          19       0.14      0.12      0.13        42

    accuracy                           0.29       811
   macro avg       0.29      0.29      0.29       811
weighted avg       0.29      0.29      0.29       811

Micro AUC: 0.7541, Macro AUC: 0.7527
```

-

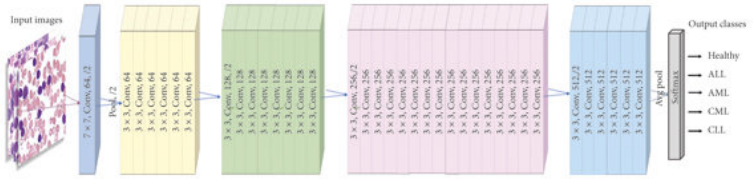


-

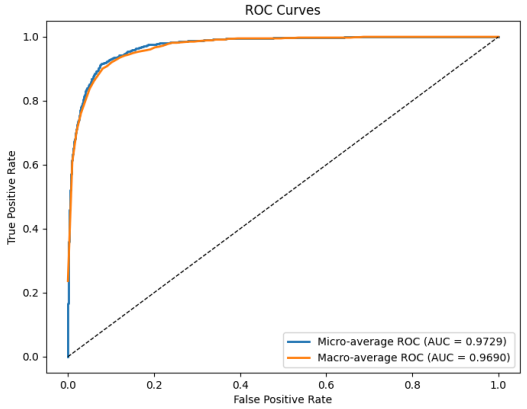


-
    - Pros:
        - No pros compared to the other models cause it is trained from scratch
    - Cons:
        - needs a lot of data and training time
        - It’s generalization on small data is poor (overfitting)
- ResNet34:
    - Step-by-step explanation:
        - Input: Accepts a RGB image resized to 224x224 pixels.
        - Convolutional Blocks:
            - Block 1: 2 convolutional layers (3x3 filters, 64 channels) followed by max pooling (2x2).
            - Block 2: 2 convolutional layers (3x3 filters, 128 channels) followed by max pooling.
            - Block 3: 4 convolutional layers (3x3 filters, 256 channels) followed by max pooling.
            - Block 4: 4 convolutional layers (3x3 filters, 512 channels) followed by max pooling.
            - Block 5: 4 convolutional layers (3x3 filters, 512 channels) followed by max pooling.
            - Each convolution uses ReLU activation; pooling reduces spatial dimensions by half.
        - Fully Connected Layers:
            - Flatten the output from Block 5.
            - Layer 1: 4096 neurons, ReLU.
            - Layer 2: 4096 neurons, ReLU.
            - Layer 3: 20neurons (for 20 classes), softmax

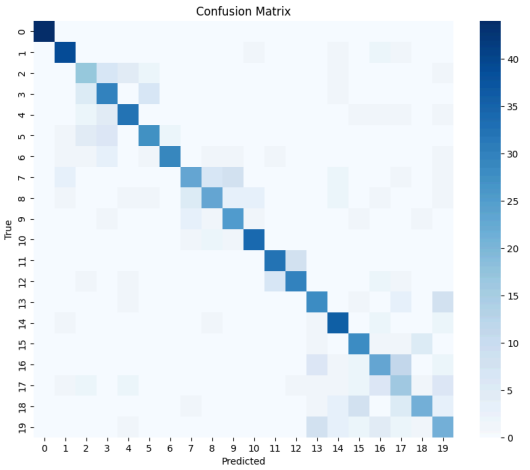
- Architecture Diagram:



- References:
  - https://www.researchgate.net/publication/373572554_The_Application_of_ResNet-34_Model_Integrating_Transfer_Learning_in_the_Recognition_and_Classification_of_Overseas_Chinese_Frescoes
  - https://arxiv.org/abs/1512.03385
- Results after transfer learning on our data:
  -
```
Epoch [10/10] Train Loss: 0.0915 Train Acc: 0.9985 val Loss: 0.3634 val Acc: 0.9314
```
  -
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        44
           1       0.83      0.89      0.86        44
           2       0.53      0.53      0.53        32
           3       0.59      0.70      0.64        43
           4       0.76      0.76      0.76        42
           5       0.71      0.68      0.69        40
           6       0.94      0.74      0.83        39
           7       0.70      0.51      0.59        45
           8       0.66      0.56      0.61        41
           9       0.66      0.76      0.70        33
          10       0.87      0.89      0.88        38
          11       0.80      0.80      0.80        40
          12       0.77      0.71      0.74        42
          13       0.61      0.68      0.64        41
          14       0.71      0.84      0.77        43
          15       0.62      0.78      0.69        36
          16       0.55      0.51      0.53        45
          17       0.38      0.41      0.40        39
          18       0.72      0.50      0.59        42
          19       0.45      0.50      0.47        42

    accuracy                           0.69       811
   macro avg       0.69      0.69      0.69       811
weighted avg       0.69      0.69      0.69       811

Micro AUC: 0.9729, Macro AUC: 0.9690
```
  -



  -

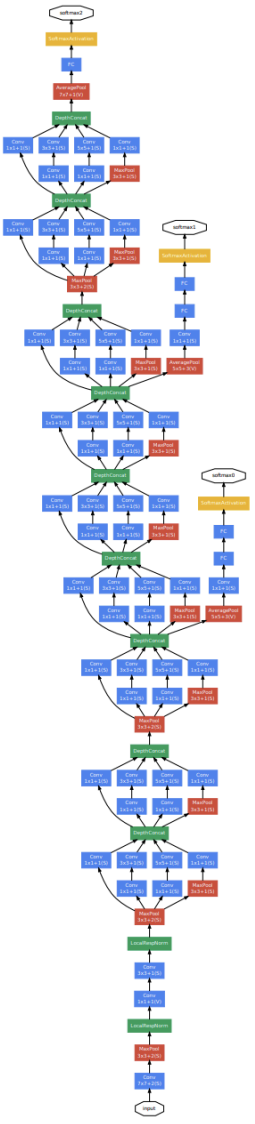


-
- Pros:
    - Small model size (parameters)
    - light-weight
- Cons:
    - Poor Generalization as it overfits the training and validation data but does not perform well on the test data
    - Due overfitting the validation it has high AUC which means it predict wrong with high confidence on the test data

- Inception v1:
    - Step-by-step explanation:
        - Input: Accepts an RGB image resized to 224x224 pixels.
        - Initial Layers:
            - Convolutional layer: 7x7 filters, 64 channels, stride 2, ReLU.
            - Max pooling: 3x3, stride 2.
            - Local response normalization.
            - Convolutional layer: 1x1, 64 channels, ReLU.
            - Convolutional layer: 3x3, 192 channels, ReLU.
            - Local response normalization.
            - Max pooling: 3x3, stride 2.
        - Inception Modules (9 modules stacked; each has parallel branches: 1x1 conv, 3x3 conv after 1x1 reduction, 5x5 conv after 1x1 reduction, 3x3 max pooling after 1x1 projection; outputs concatenated):
            - Modules 3a, 3b: At 256 output channels.
            - Max pooling: 3x3, stride 2.
            - Modules 4a-4e: At 480-1024 output channels.
            - Max pooling: 3x3, stride 2.
            - Modules 5a, 5b: At 1024 output channels.

        - ReLU activation throughout.

    - Auxiliary Classifiers: Two added at intermediate layers (after 4a and 4d) for gradient flow; each includes average pooling, 1x1 conv, fully connected layers, and softmax.
    - Final Layers:
        - Average pooling: 7x7.
        - Dropout (40%).
        - Fully connected layer: 20 neurons (for 20 classes), softmax for probabilities.
- Architecture Diagram:
    -

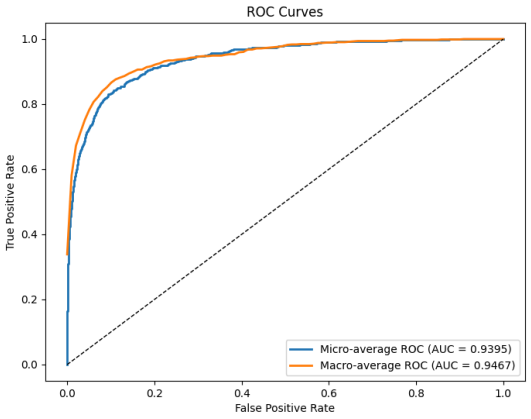


- Reference: https://arxiv.org/abs/1409.4842
- Results after Transfer learning on our data:
    -

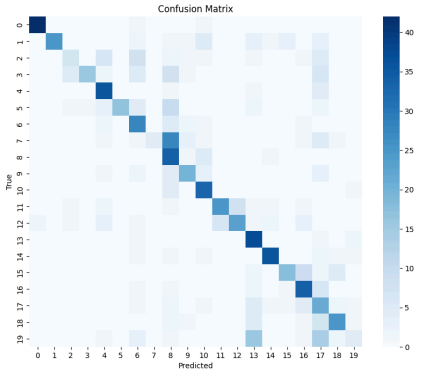
```
Train Loss (MAE): 0.0747 | Train Acc: 97.90%
Val Loss (MAE):   0.6738 | Val Acc:   82.62%
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        44
           1       1.00      0.57      0.72        44
           2       0.47      0.22      0.30        32
           3       0.94      0.37      0.53        43
           4       0.63      0.86      0.73        42
           5       1.00      0.42      0.60        40
           6       0.47      0.72      0.57        39
           7       1.00      0.09      0.16        45
           8       0.32      0.83      0.47        41
           9       0.67      0.61      0.63        33
          10       0.63      0.87      0.73        38
          11       0.81      0.62      0.70        40
          12       0.74      0.55      0.63        42
          13       0.49      0.90      0.63        41
          14       0.86      0.84      0.85        43
          15       0.78      0.50      0.61        36
          16       0.61      0.76      0.67        45
          17       0.26      0.54      0.35        39
          18       0.69      0.60      0.64        42
          19       0.40      0.10      0.15        42

    accuracy                           0.60       811
   macro avg       0.69      0.60      0.58       811
weighted avg       0.69      0.60      0.59       811

Micro AUC: 0.9395, Macro AUC: 0.9467
```

-



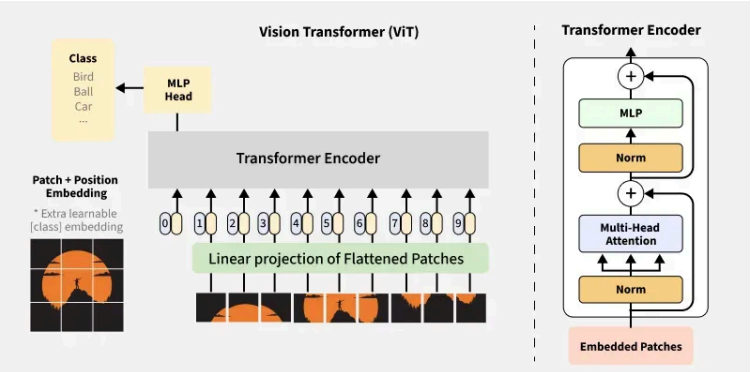
-



-

- Pros:
    - Very small and efficient model (only 6 million parameters)
- Cons:
    - Overfits over small datasets

- VIT:
    - Step-by-step explanation:
        - Input: RGB image resized to 224x224 pixels.
        - Patch Embedding:
            - Split image into fixed-size patches (e.g., 16x16).
            - Flatten each patch into a vector.
            - Linear projection to embed into D dimensions (e.g., 768).
            - Add learnable class token and positional embeddings.
        - Transformer Encoder (12 layers for base model):
            - Multi-head self-attention: Computes relationships between all patches.
            - MLP block: Two linear layers with GELU activation.
            - Layer normalization and residual connections after each sub-block.
        - Final Layers:
            - Take output of class token.
            - MLP head: Fully connected layer for classification, softmax for probabilities.

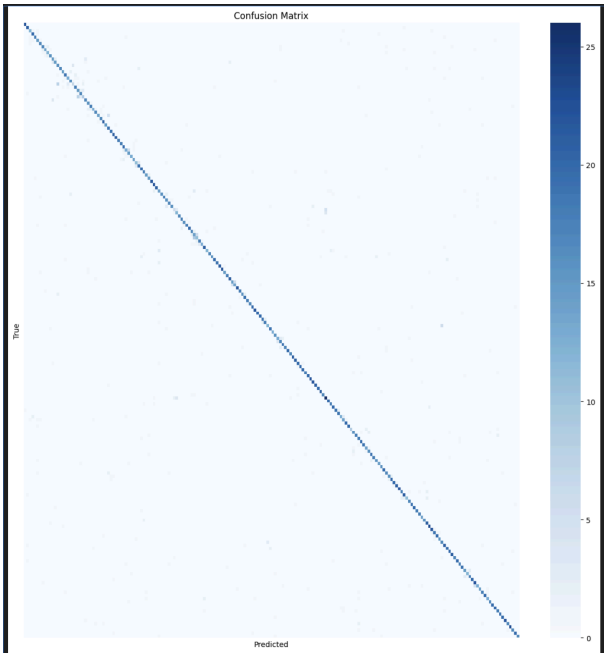
        - Architecture Diagram:
        -


        - References: https://arxiv.org/abs/2010.11929
        - Results after training on the full dataset:
        -
```
--- Val Loss: 0.5300 | Acc: 85.92% ---
Training Complete. Total time: 89.7 mins
```

- Classification Report:

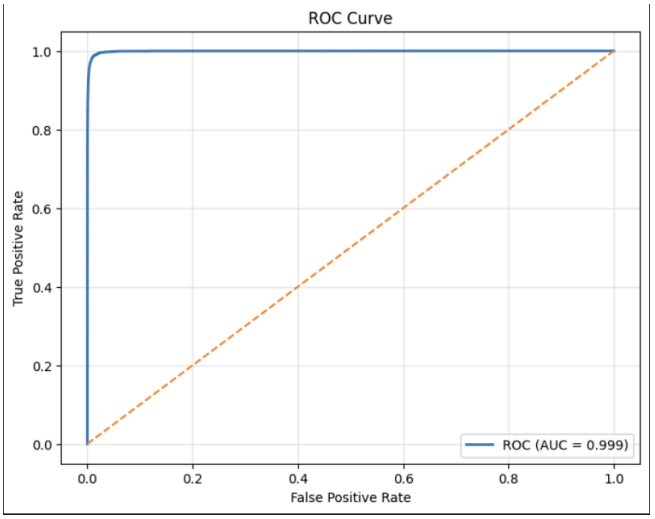
	precision	recall	f1-score	support
0	0.9565	1.0000	0.9778	22
1	0.8400	0.9545	0.8936	22
2	0.7500	0.7500	0.7500	16
3	0.8333	0.9091	0.8696	22
4	0.9474	0.8571	0.9000	21
5	0.6818	0.7500	0.7143	20
6	0.8095	0.8947	0.8500	19
7	0.8095	0.7391	0.7727	23
8	0.7778	0.7000	0.7368	20
9	0.7500	0.7500	0.7500	16
10	0.8421	0.8421	0.8421	19
11	0.6522	0.7500	0.6977	20
12	0.8333	0.7143	0.7692	21
13	0.5714	0.8000	0.6667	20
14	1.0000	0.7727	0.8718	22
15	0.8333	0.8333	0.8333	18
16	0.7917	0.8261	0.8085	23
17	0.6842	0.6842	0.6842	19
18	0.8000	0.7619	0.7805	21
19	0.5833	0.3333	0.4242	21
20	0.8571	0.7826	0.8182	23
...				
accuracy			0.8558	4021
macro avg	0.8607	0.8552	0.8544	4021
weighted avg	0.8617	0.8558	0.8553	4021

-



-

-



  - Pros:
    - High accuracy on the full dataset.
  - Cons:
    - No cons as an architecture but in our case it is slightly overfitting
- Best model:
  - VIT as the attention makes the model capture the real important details in the images as our problem is fine-grained classification.