Multimodal Data Fusion for Image Classification

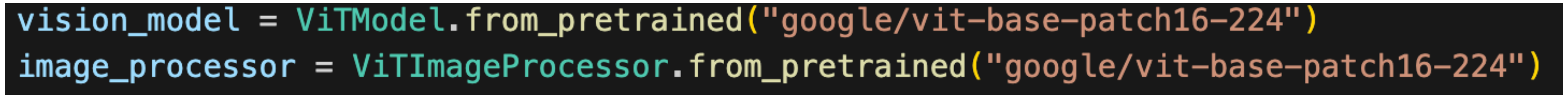
Stefano Ruggiero

Project Goal

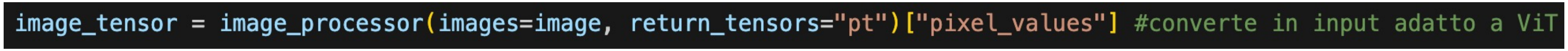
- The goal of this project was to classify objects in the NuScenes dataset (mini version with 400 samples and 18 categories) by combining data from two modalities: **LiDAR point clouds** and **camera images**.

Approach

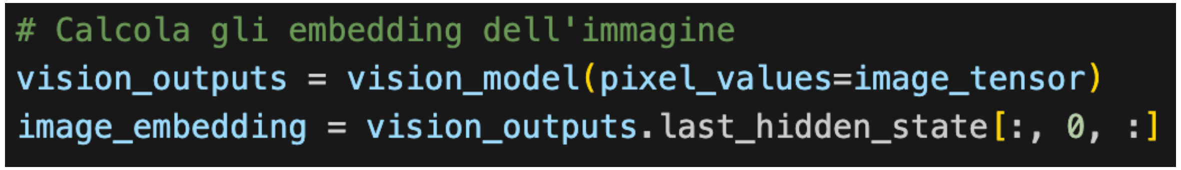
- A **Vision Transformer (ViT)** (https://huggingface.co/google/vit-base-patch16-224) to process image data: pre-trained on ImageNet-21k (14 million images, 21,843 classes), and fine-tuned on ImageNet 2012 (1 million images, 1,000 classes). Images are presented to the model as a sequence of fixed-size patches (resolution 16x16), which are linearly embedded. One also adds a [CLS] token to the beginning of a sequence to use it for classification tasks. One also adds absolute position embeddings before feeding the sequence to the layers of the Transformer encoder.



```
vision_model = ViTModel.from_pretrained("google/vit-base-patch16-224")
image_processor = ViTImageProcessor.from_pretrained("google/vit-base-patch16-224")
```



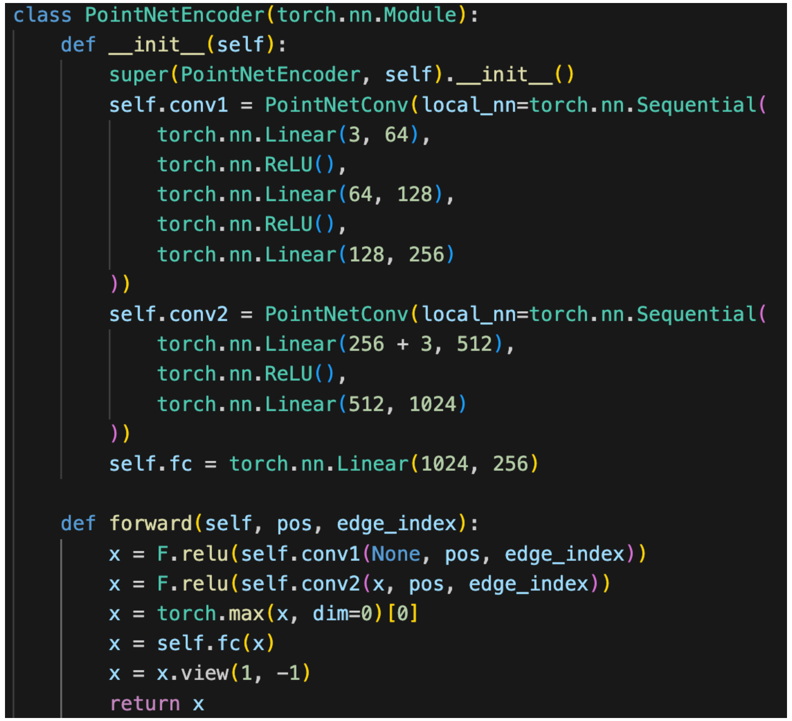
```
image_tensor = image_processor(images=image, return_tensors="pt")["pixel_values"] #converte in input adatto a ViT
```



```
# Calcola gli embedding dell'immagine
vision_outputs = vision_model(pixel_values=image_tensor)
image_embedding = vision_outputs.last_hidden_state[:, 0, :]
```

Approach

- **PointNet** (PointNetConv) to encode 3D LiDAR data: The PointNetConv set layer from the PointNet and PointNet papers. Utilizza convoluzioni basate su grafi (**Graph Neural Networks**). Knn_graph computes graph edges to the nearest k points (matrice).

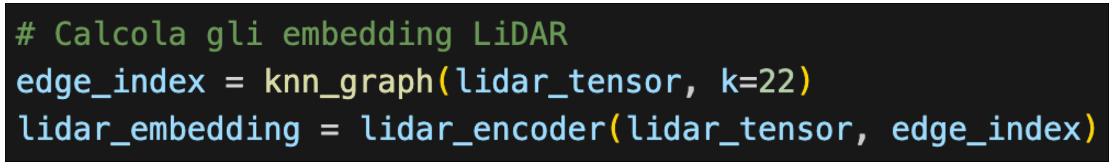


```python
class PointNetEncoder(torch.nn.Module):
    def __init__(self):
        super(PointNetEncoder, self).__init__()
        self.conv1 = PointNetConv(local_nn=torch.nn.Sequential(
            torch.nn.Linear(3, 64),
            torch.nn.ReLU(),
            torch.nn.Linear(64, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, 256)
        ))
        self.conv2 = PointNetConv(local_nn=torch.nn.Sequential(
            torch.nn.Linear(256 + 3, 512),
            torch.nn.ReLU(),
            torch.nn.Linear(512, 1024)
        ))
        self.fc = torch.nn.Linear(1024, 256)

    def forward(self, pos, edge_index):
        x = F.relu(self.conv1(None, pos, edge_index))
        x = F.relu(self.conv2(x, pos, edge_index))
        x = torch.max(x, dim=0)[0]
        x = self.fc(x)
        x = x.view(1, -1)
        return x
```

https://pytorch-geometric.readthedocs.io/en/2.6.0/generated/torch_geometric.nn.pool.knn_graph.html

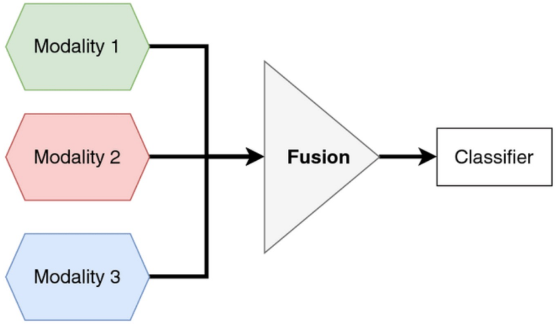
https://pytorch-geometric.readthedocs.io/en/stable/generated/torch_geometric.nn.conv.PointNetConv.html

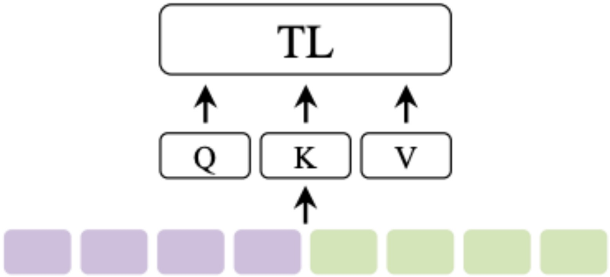


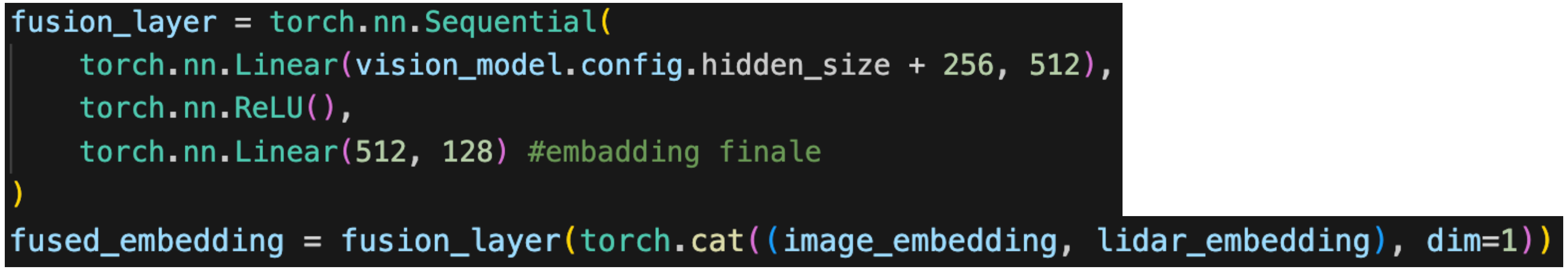
```python
# Calcola gli embedding LiDAR
edge_index = knn_graph(lidar_tensor, k=22)
lidar_embedding = lidar_encoder(lidar_tensor, edge_index)
```

Approach

- An **early fusion strategy** (concatenation) that combines features from both modalities into a single representation: a network to get an embadding of the concatenation of the two rapresentation.



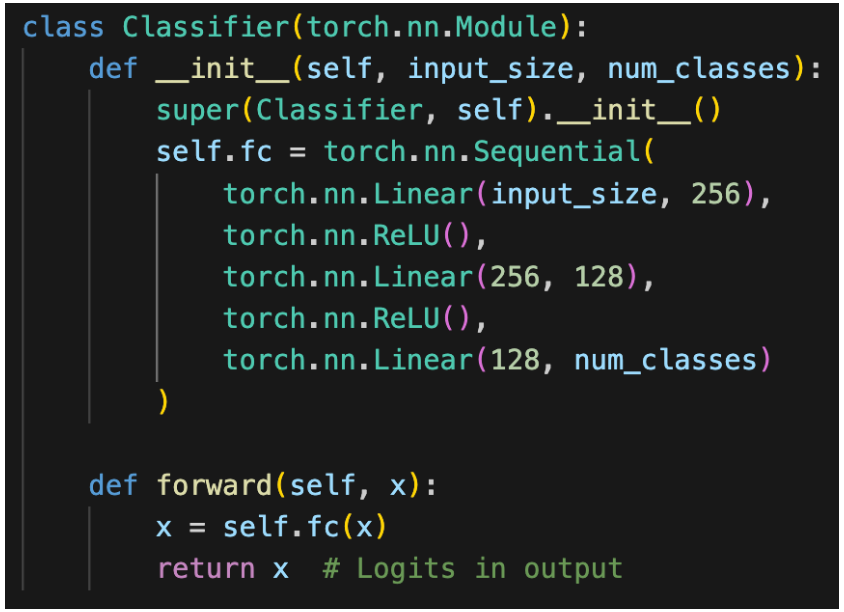




```
fusion_layer = torch.nn.Sequential(
    torch.nn.Linear(vision_model.config.hidden_size + 256, 512),
    torch.nn.ReLU(),
    torch.nn.Linear(512, 128) #embadding finale
)
fused_embedding = fusion_layer(torch.cat((image_embedding, lidar_embedding), dim=1))
```

Approach

- **Classification** :with neural network



```python
class Classifier(torch.nn.Module):
    def __init__(self, input_size, num_classes):
        super(Classifier, self).__init__()
        self.fc = torch.nn.Sequential(
            torch.nn.Linear(input_size, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 128),
            torch.nn.ReLU(),
            torch.nn.Linear(128, num_classes)
        )

    def forward(self, x):
        x = self.fc(x)
        return x  # Logits in output
```

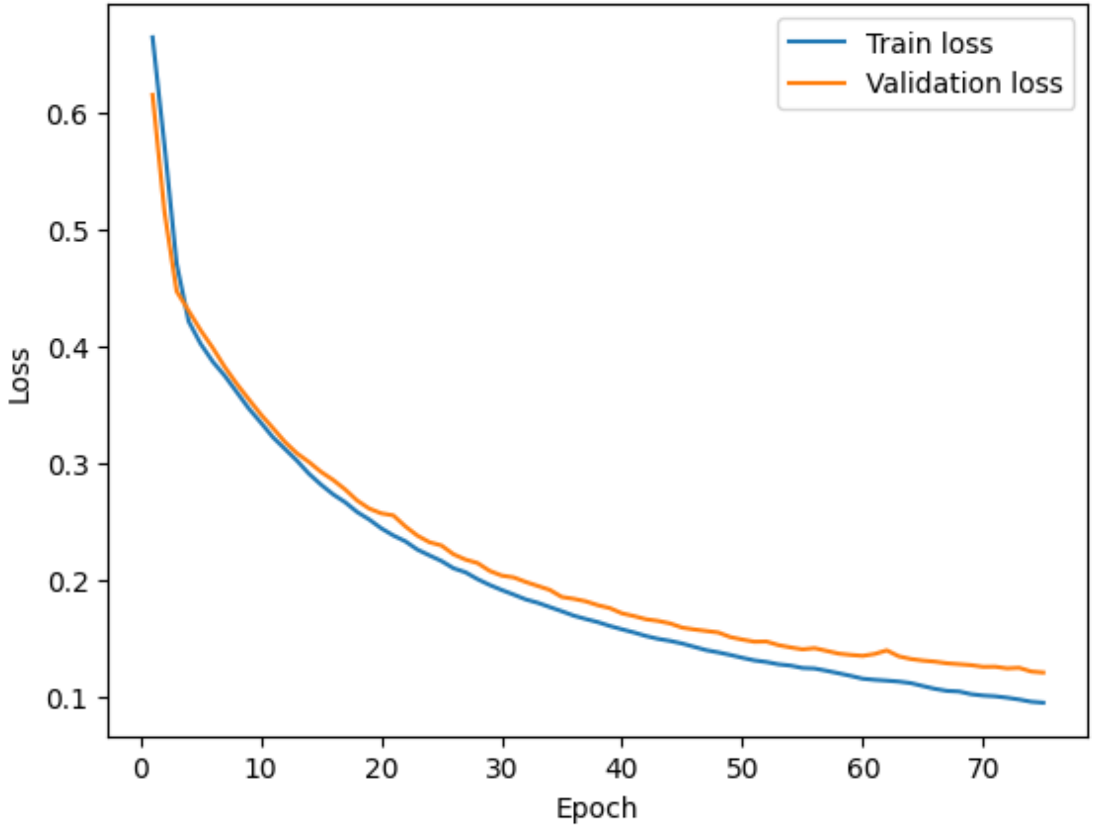
Run the code

- **Dataset Loading & Preprocessing**
- **Fusion**
- **Classification**
- **Evaluation (F1- score)**

Result (mean on 1000 runs on 1000 split)

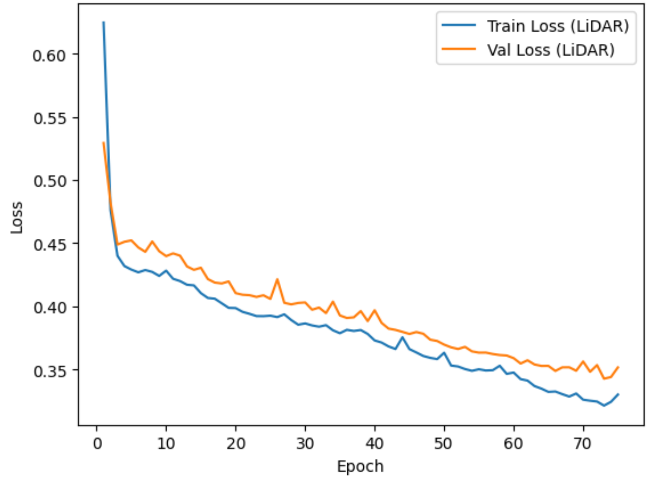
- **Fused embeddings**
  F1 Score: 0.914

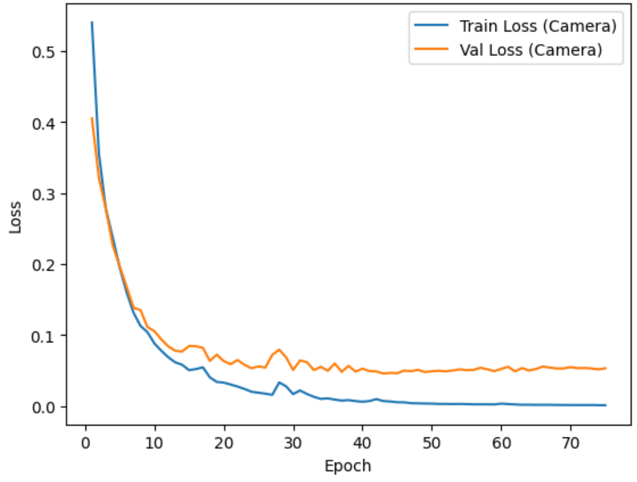
- Is good a result ?



Results (mean on 1000 runs on 1000 split)

- **Camera embeddings** achieved the highest F1 score (0.96), highlighting the effectiveness of ViT for image data.
- **Fused embeddings** performed well (F1 Score: 0.914) but slightly underperformed compared to camera-only embeddings.
- **LiDAR embeddings** scored the lowest (F1 Score: 0.74).





Challenges

- LiDAR data lacks the granularity of visual data, which limited its standalone performance. Problably caused by: not enought detail of graph (k), the encoder or the few data.
- The computational cost of processing multimodal data.

Takeaways

- **Encode images with ViT**
- **Encode point clouds with PointNetConv**
- **Make early fusion by neaural network**
- **Make predictions with fused data by neaural network**