

YOGA POSES CLASSIFICATION

Classify yoga poses using a pre-trained Transformer with Graph Neural Network layers to jointly process images and pose graphs using a <CLS> token for accurate prediction

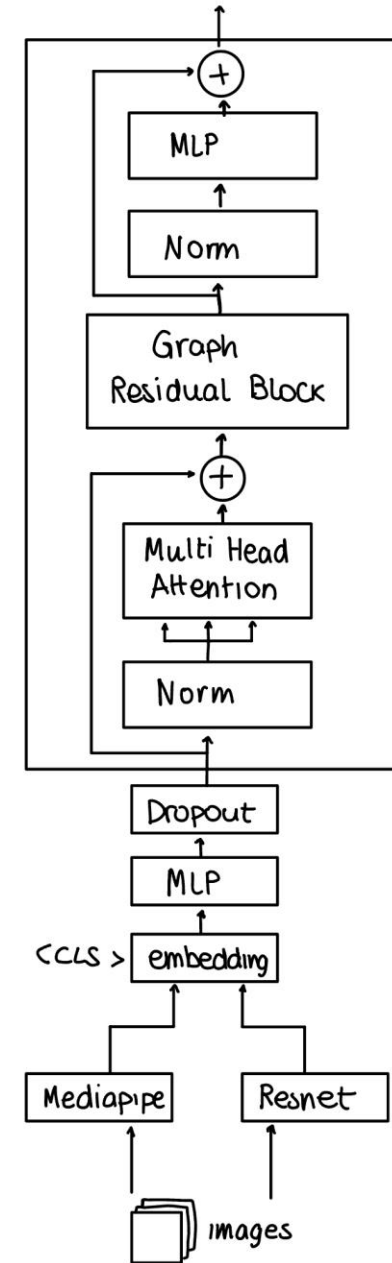
ARCHITECTURE COMPONENTS

MEDIAPIPE

RESNET

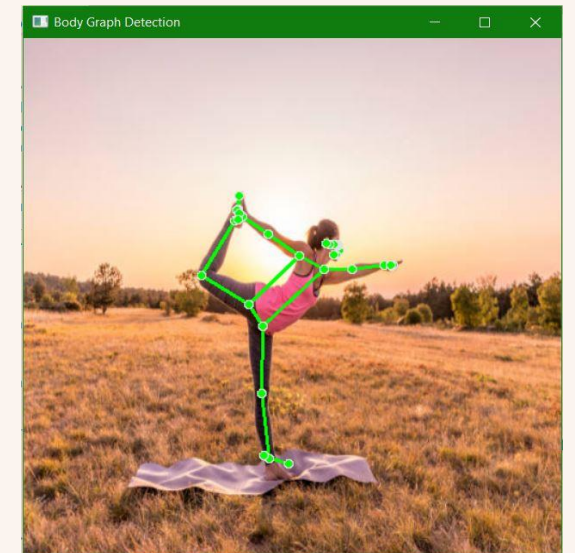
GRAPH
RESIDUAL
BLOCK

BERT

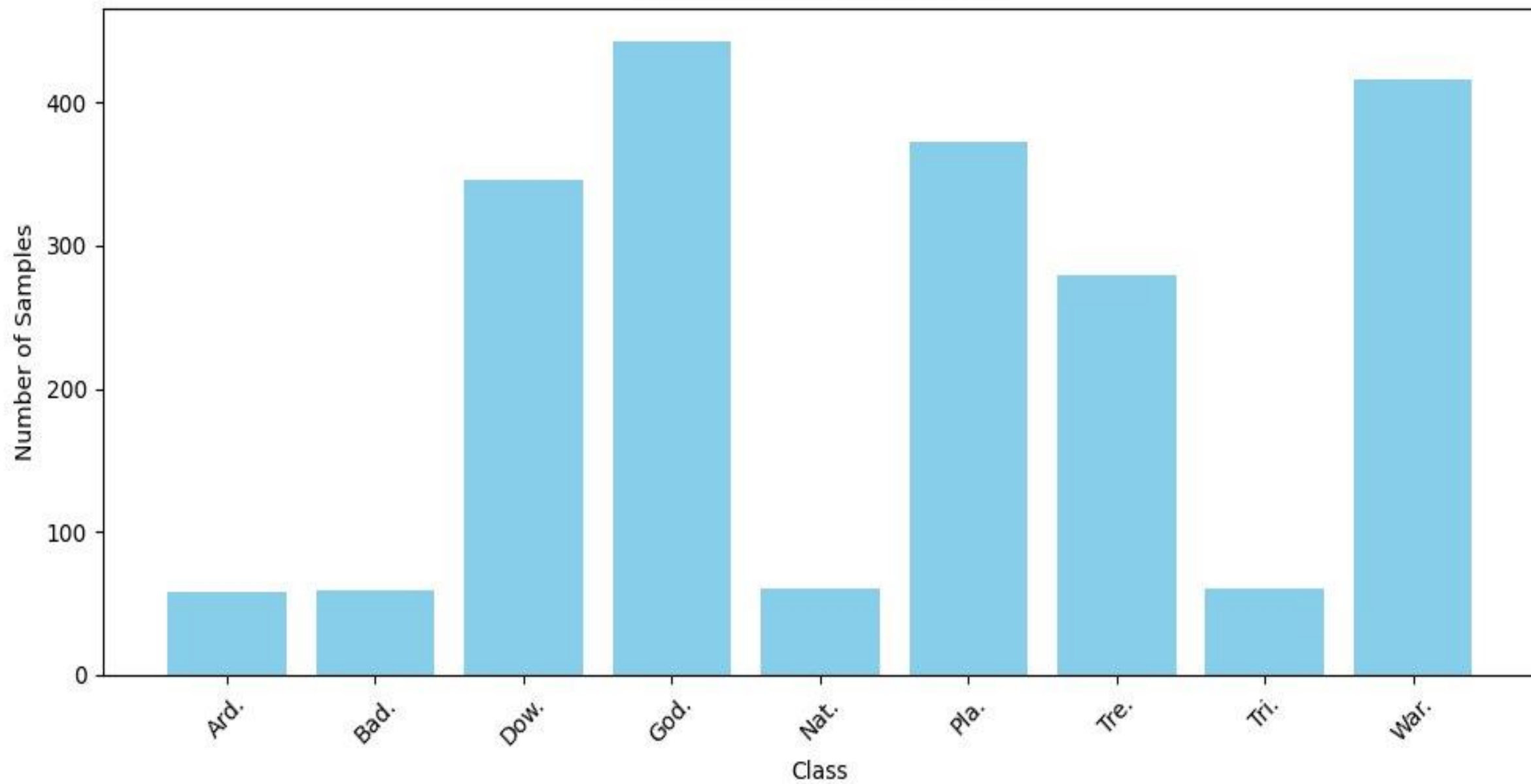


DATASET

- Loaded image data from class-labeled directories.
- Normalized images to RGB and resizes them.
- Extracted pose landmarks using MediaPipe.
- Saved processed images and keypoints to disk.
- Converted images and landmarks to PyTorch tensors.
- Applied one-hot encoding to class labels.
- Shuffled and splits data into training and test sets.
- Created separate datasets for images, keypoints, and edges.
- Returned batched data loaders for model training.



Class Distribution in Dataset



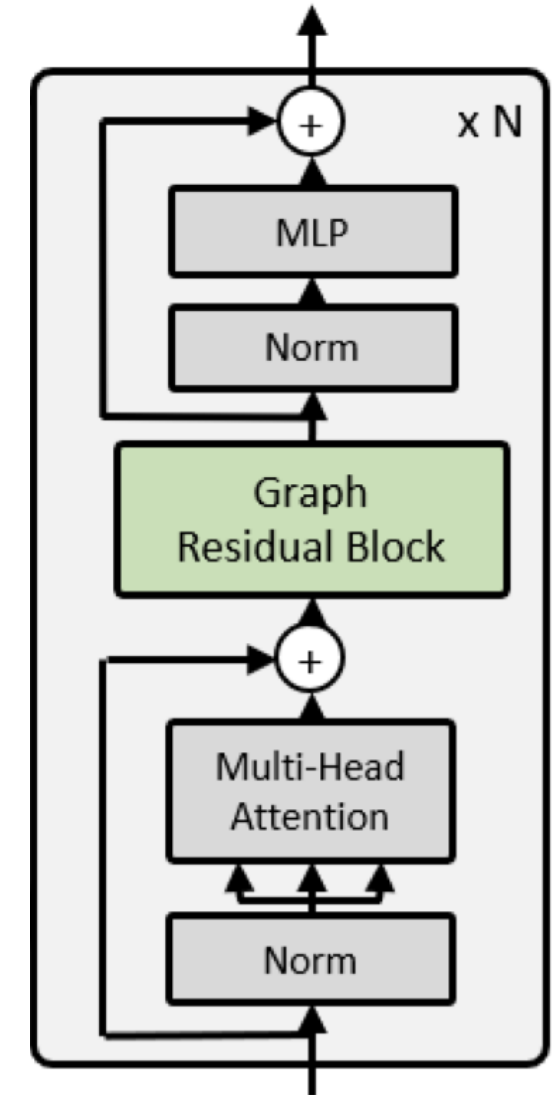
EMBEDDINGS

- Combines three sources of information for each sample:
- **CNN visual features** from ResNet
- **3D landmark coordinates** (x, y, z, visibility)
- **Skeleton graph structure** as a flattened adjacency matrix
- All components are **concatenated** into a single feature vector
- A **[CLS] token** is prepended to the sequence for transformer-based processing
- Final tensor is **replicated** to form a sequence input compatible with BERT+GNN layers

```
#print(deviceAdjMat[em].device)
emb= torch.concat((elem,concatCoor(vt),adjMat))
emb = torch.stack([emb for x in range(mI+1)])
emb = torch.concat((clsToken,emb))
#print("shape")
```

BERT

- Uses a pre-trained BERT-base model as the core transformer encoder
- Replaces standard BERT encoder layers with custom BertGraphEncoder modules
- Each BertGraphEncoder injects graph structure via a GraphResidualBlock into attention output
- BERT attention captures global context, GCN layers capture local graph relations

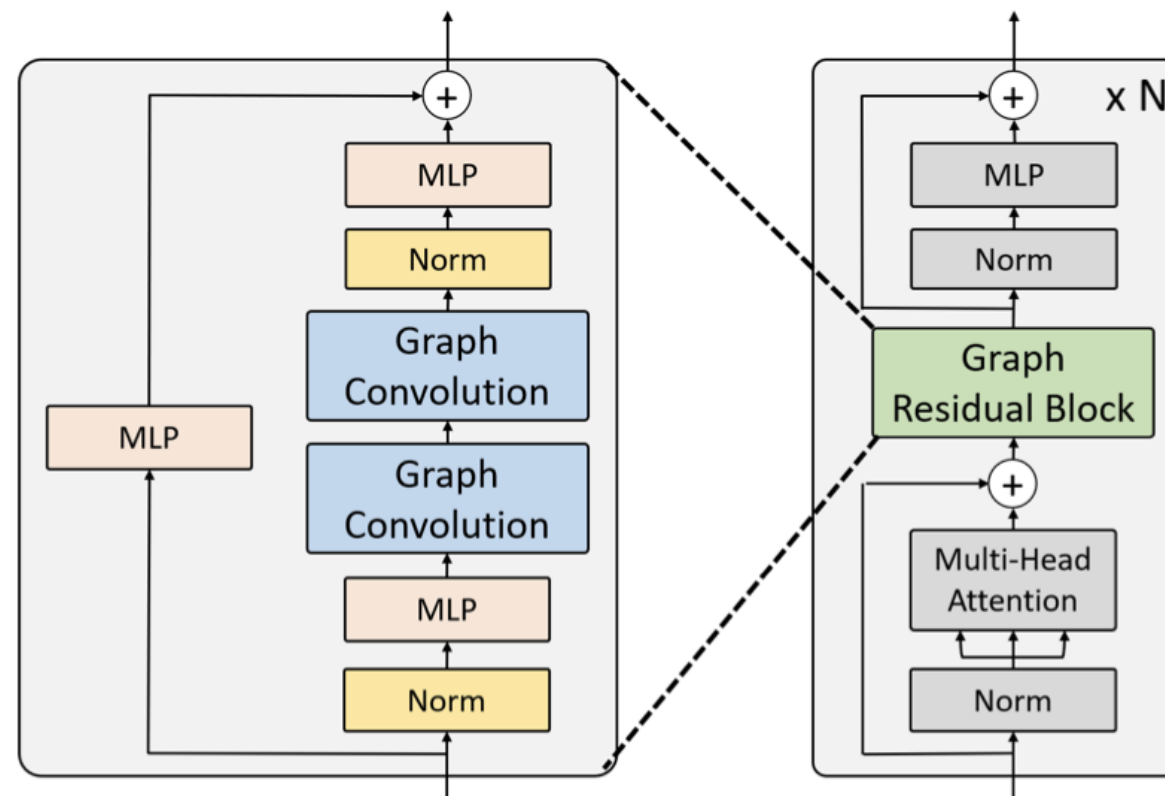


- Starts with LayerNorm and MLP for feature transformation
- Applies two GCNConv layers with GELU activations for message passing
- Includes a residual connection with its own MLP to preserve input information

```
class GraphResidualBlock(nn.Module):
    def __init__(self, in_channels, out_channels, hiddenDim):
        super().__init__()

        self.norm1 = nn.LayerNorm(hiddenDim)
        # mlp1 layer
        self.mlp1 = nn.Sequential(
            nn.Linear(hiddenDim, hiddenDim),
            nn.GELU())
        # graph conv
        self.conv1 = GCNConv(in_channels, out_channels)
        # gelu
        self.gelu1 = nn.GELU()
        # graph conv
        self.conv2 = GCNConv(in_channels, out_channels)
        # gelu
        self.gelu2 = nn.GELU()
        # norm
        self.norm2 = nn.LayerNorm(hiddenDim)
        # mlp2
        self.mlp2 = nn.Sequential(
            nn.Linear(hiddenDim, hiddenDim),
            nn.GELU())
        # + residual mlp
        self.mlp3 = nn.Sequential(
            nn.Linear(hiddenDim, hiddenDim),
            nn.GELU())
```

GRAPH RESIDUAL BLOCK



TRAINING PIPELINE

Input: (Image,
Graph) pairs per sample

Early Stopping: Stops if
validation loss doesn't improve
for 3 epochs

Loss: CrossEntropyLoss with
class weighting to handle
imbalance

**Best model
checkpointing** based on
validation loss.

GRID SEARCH

```
param_grid = {  
    'lr': [1e-4, 1e-5],  
    'weight_decay': [1e-4, 1e-5],  
    'dropout_rate': [0.2, 0.3, 0.5],  
    'numLayers': [2, 4, 6]  
}
```

5 classes classification

Best parameters:

```
{'lr': 1e-05, 'weight_decay': 1e-05, 'dropout_rate': 0.5, 'numLayers': 2}
```

Best validation accuracy: 95.1814

9 classes classification

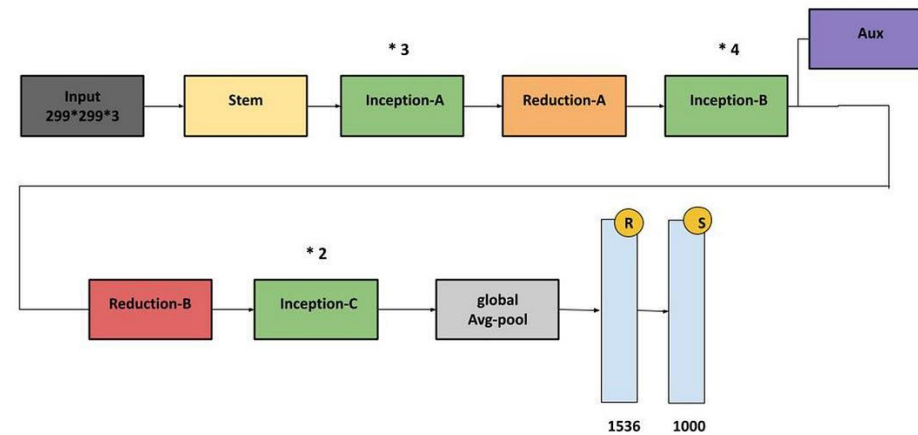
Best parameters:

```
{'lr': 1e-05, 'weight_decay': 1e-05, 'dropout_rate': 0.5, 'numLayers': 6}
```

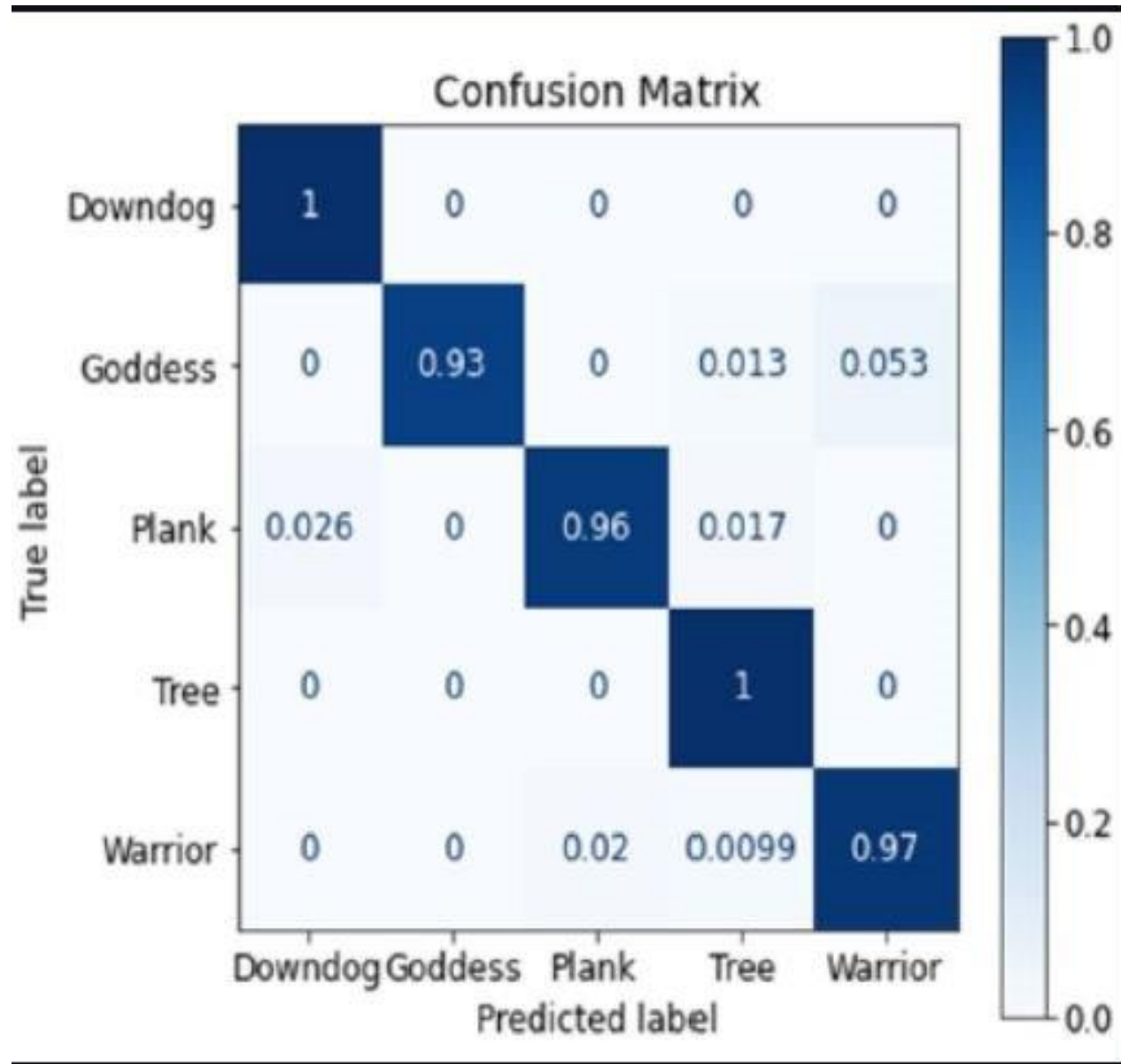
Best validation accuracy: 0.9605

YOGACONV2D & INCEPTIONV3

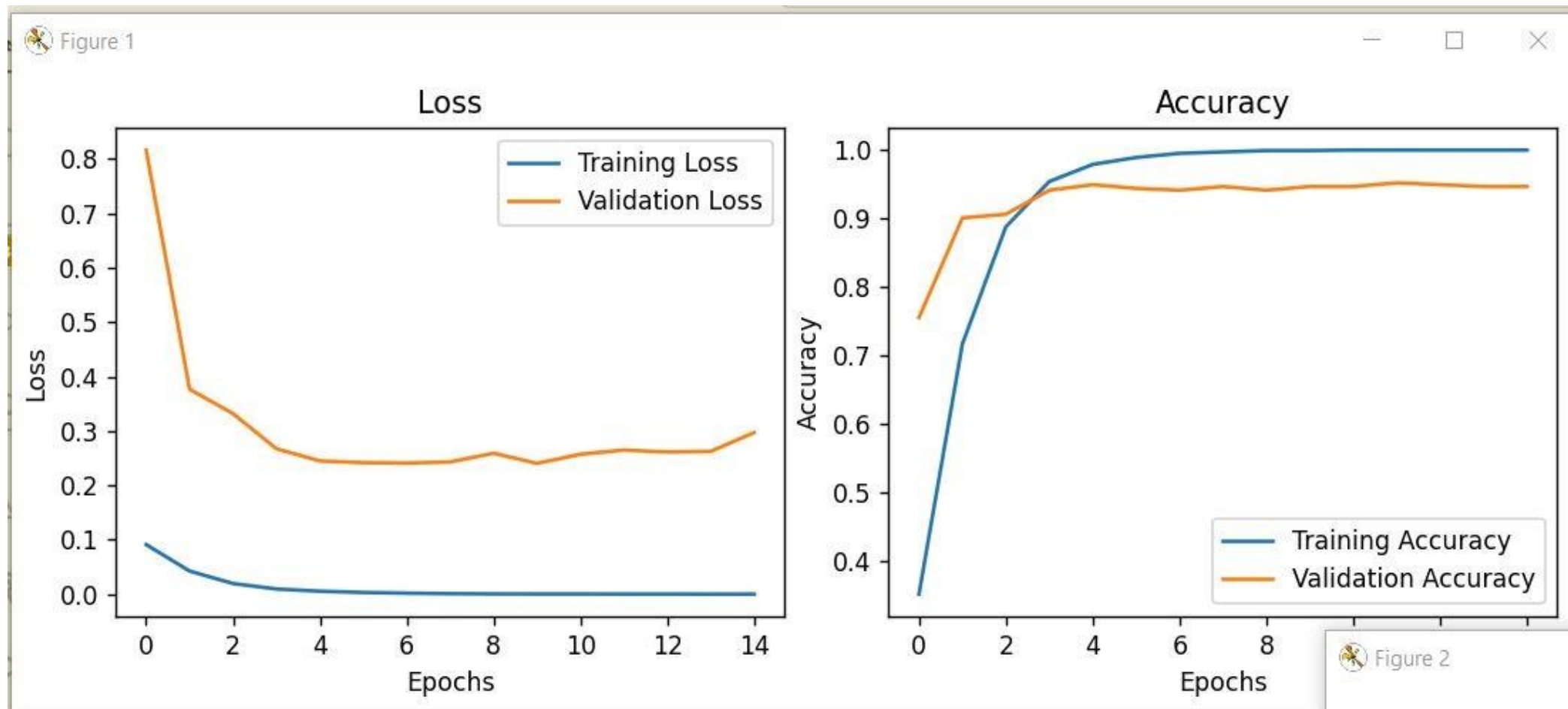
Inception V3



Model	Accuracy (Validation)	Accuracy (Test)	Precision	Recall	F1 Score
YogaConvo2d (MediaPipe)	99.62%	97.09%	0.97	0.97	0.97
YogaConvo2d (Non-MediaPipe)	89.97%	89.36%	0.89	0.88	0.89
InceptionV3 (MediaPipe)	95.09%	94.39%	0.95	0.94	0.94



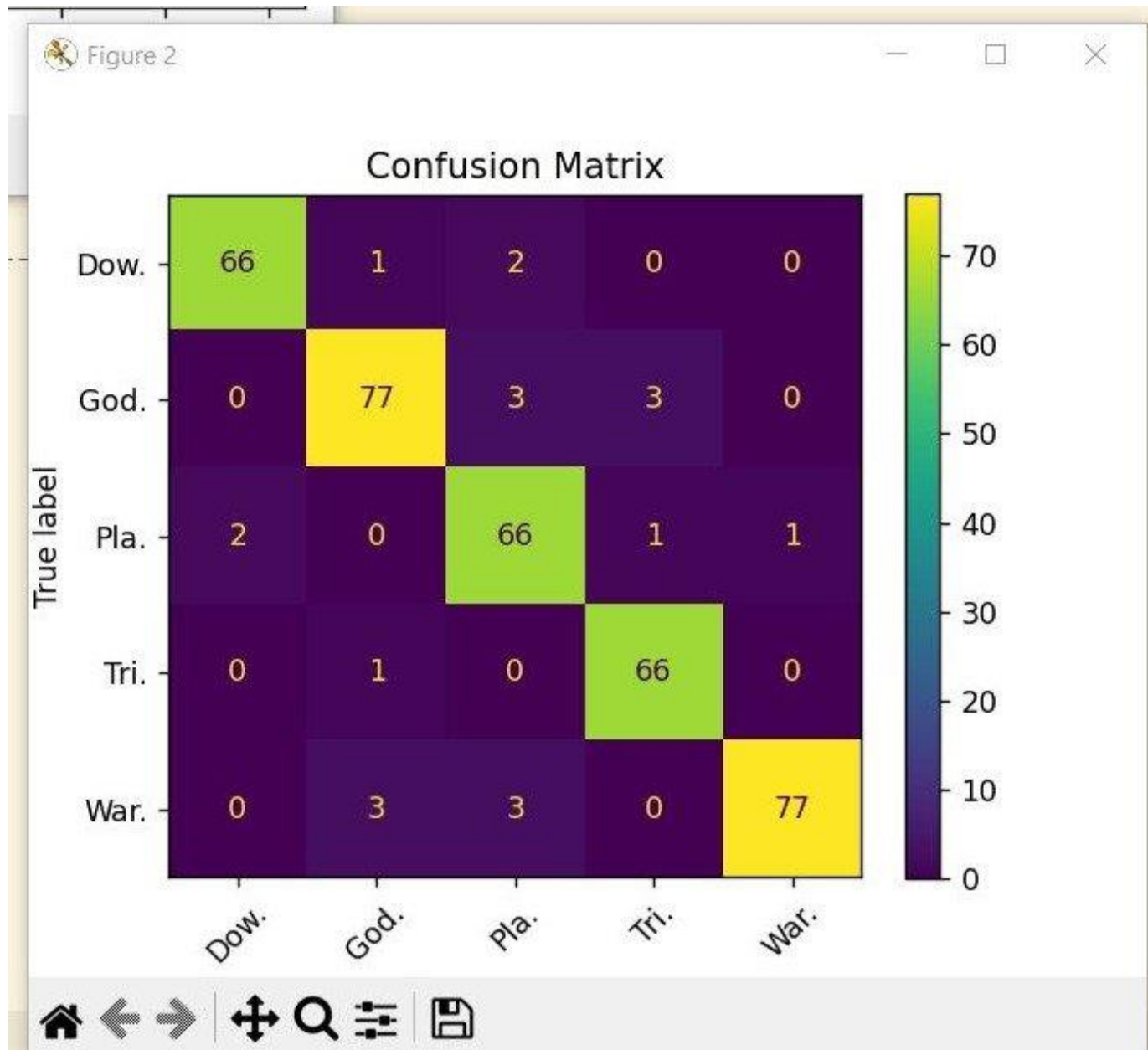
BERT + GRAPH RESIDUAL BLOCK 5

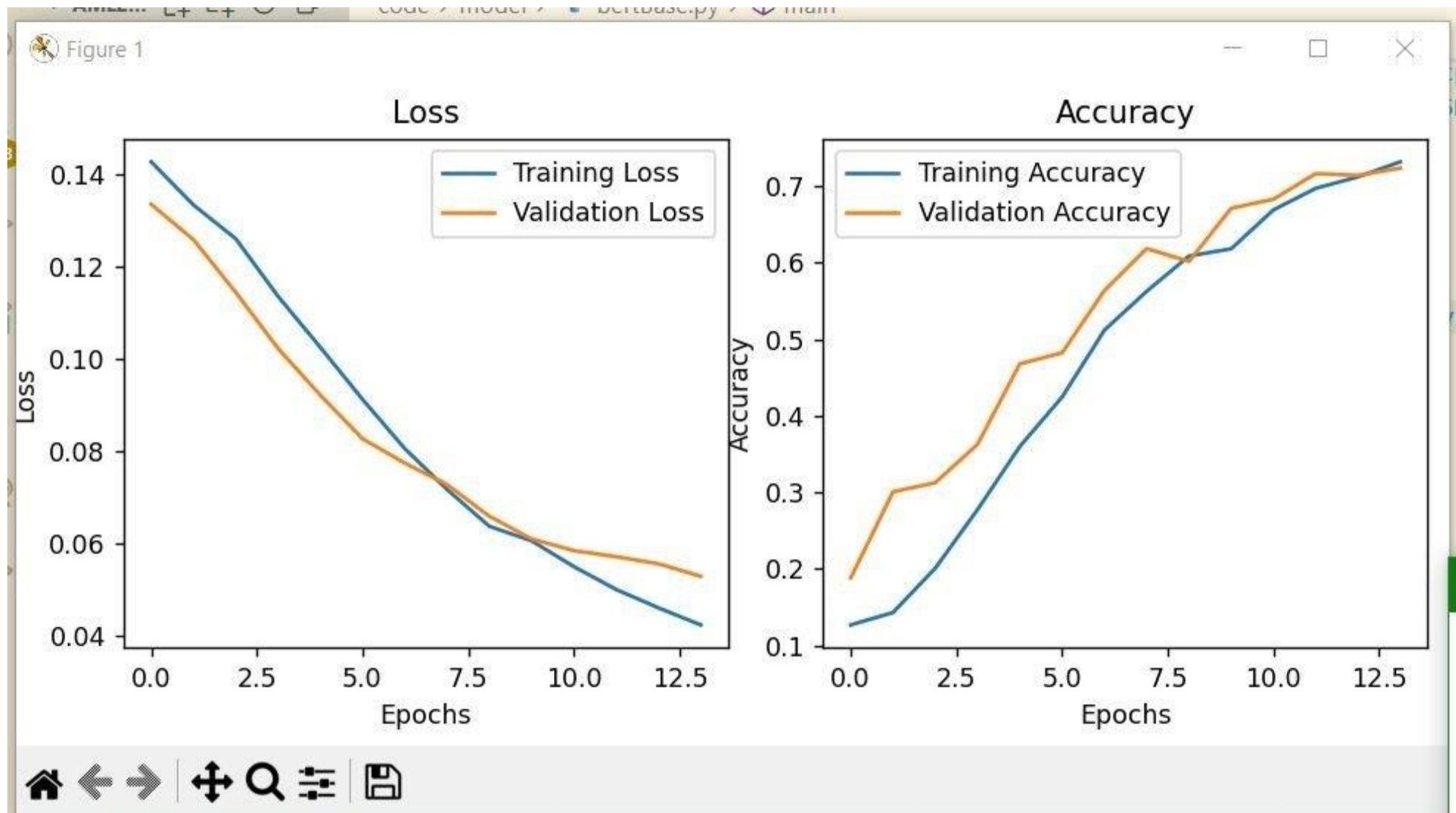


94.6236559139785

	precision	recall	f1-score	support
Dow.	0.9706	0.9565	0.9635	69
God.	0.9390	0.9277	0.9333	83
Pla.	0.8919	0.9429	0.9167	70
Tri.	0.9429	0.9851	0.9635	67
War.	0.9872	0.9277	0.9565	83
accuracy			0.9462	372
macro avg	0.9463	0.9480	0.9467	372
weighted avg	0.9474	0.9462	0.9464	372



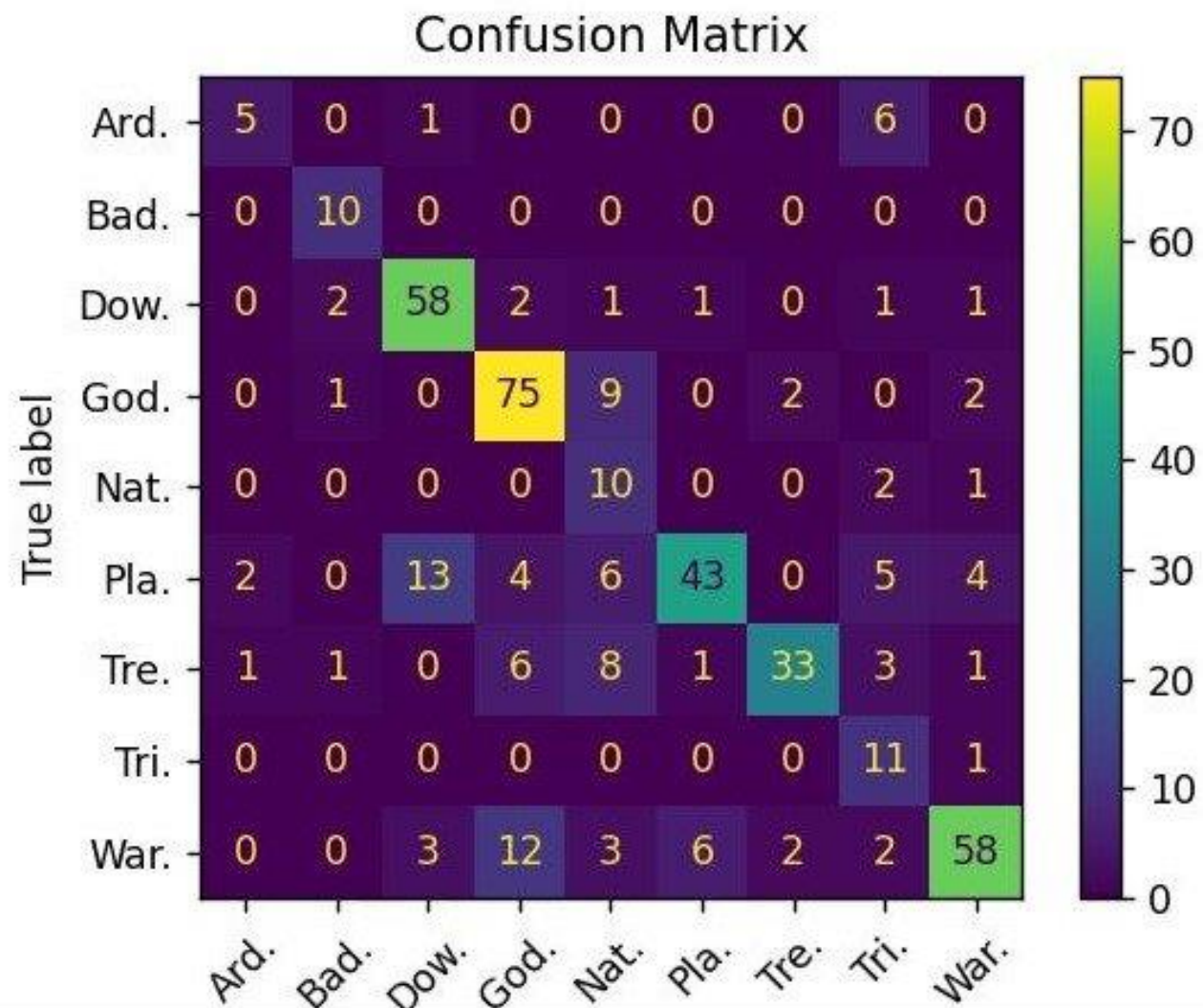




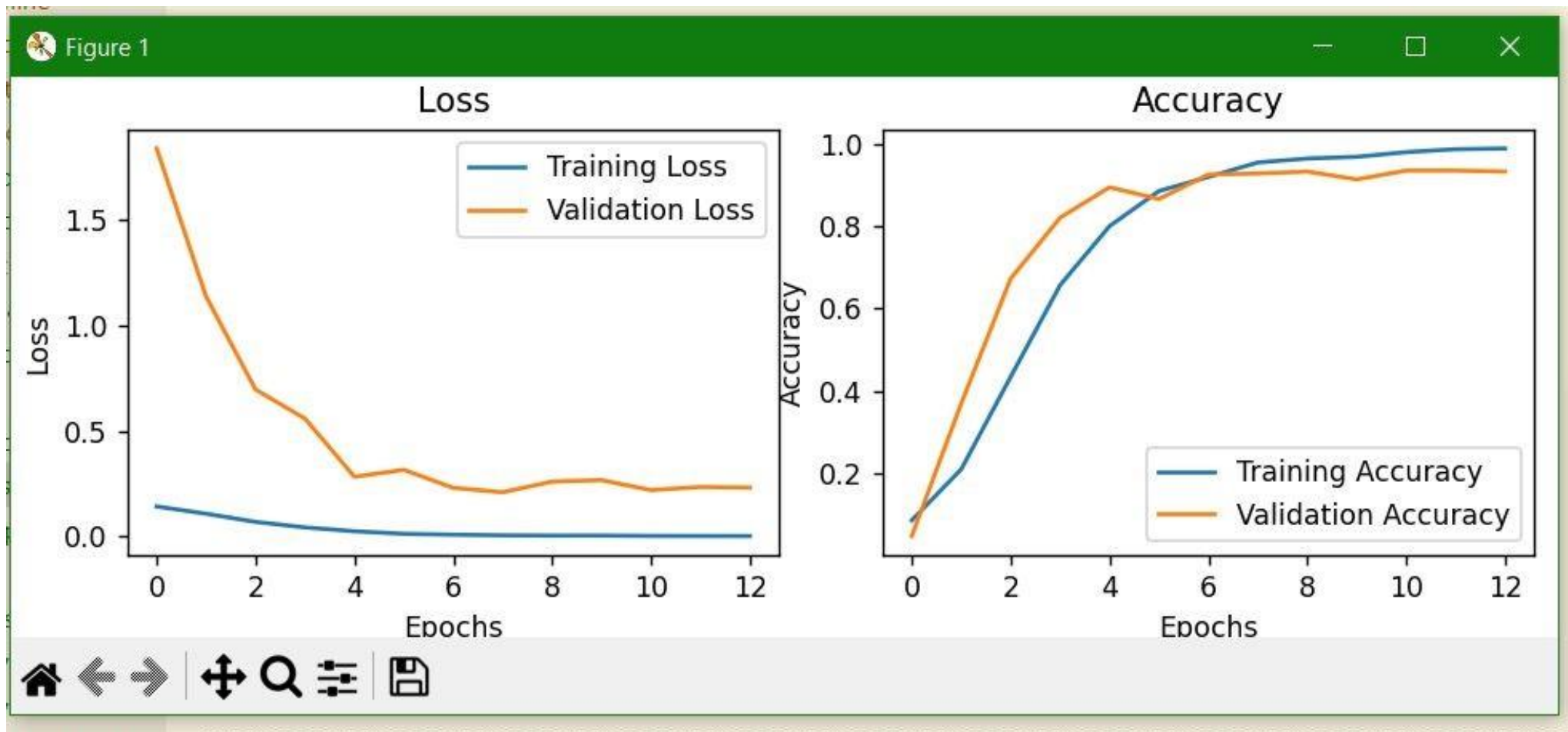
	precision	recall	f1-score	support
Ard.	0.6250	0.4167	0.5000	12
Bad.	0.7143	1.0000	0.8333	10
Dow.	0.7733	0.8788	0.8227	66
God.	0.7576	0.8427	0.7979	89
Nat.	0.2703	0.7692	0.4000	13
Pla.	0.8431	0.5584	0.6719	77
Tre.	0.8919	0.6111	0.7253	54
Tri.	0.3667	0.9167	0.5238	12
War.	0.8529	0.6744	0.7532	86
accuracy			0.7232	419
macro avg	0.6772	0.7409	0.6698	419
weighted avg	0.7815	0.7232	0.7322	419

□

Figure 2

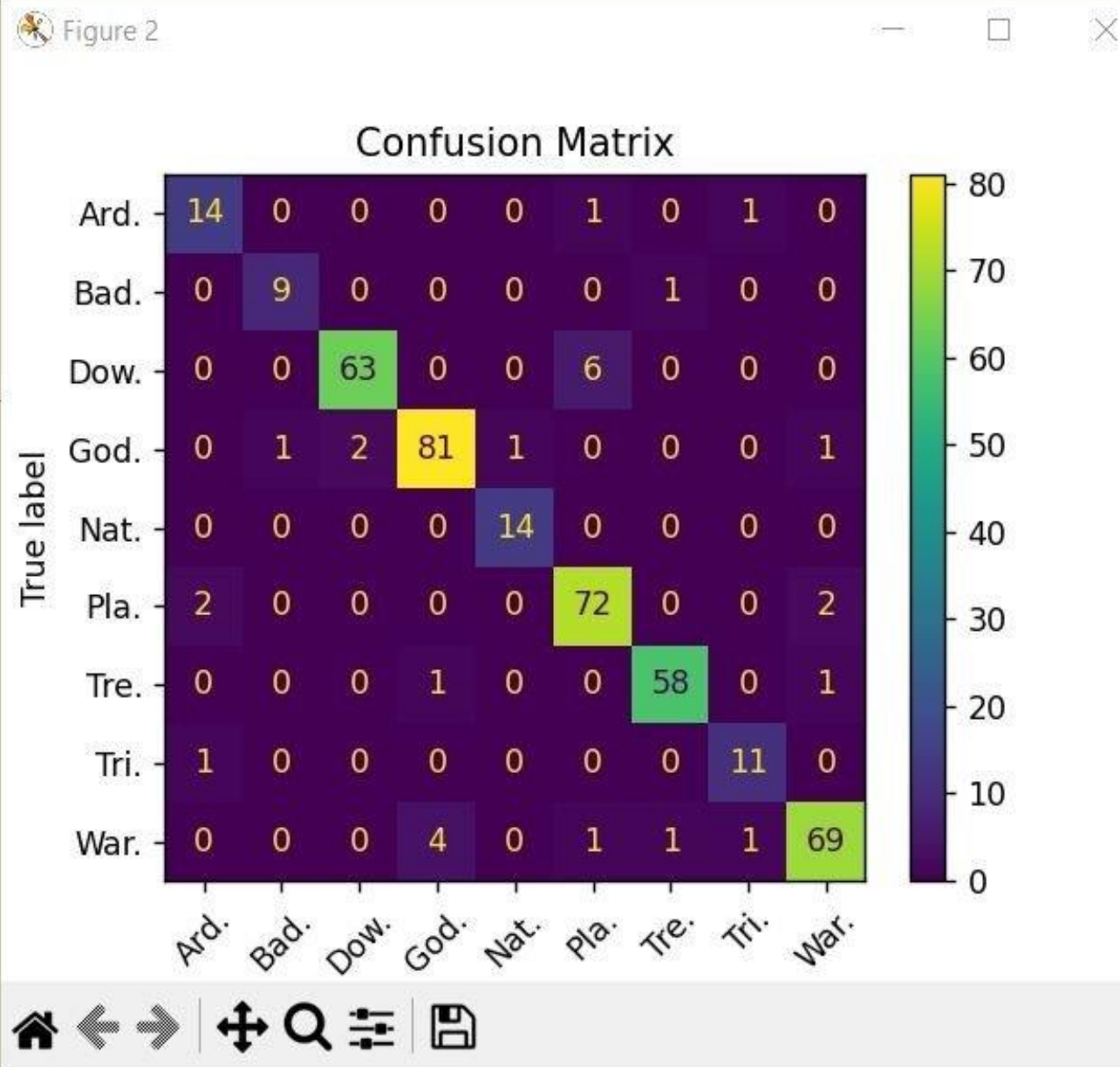



BERT + GRAPH RESIDUAL BLOCK 9



	precision	recall	f1-score	support
Ard.	0.8235	0.8750	0.8485	16
Bad.	0.9000	0.9000	0.9000	10
Dow.	0.9692	0.9130	0.9403	69
God.	0.9419	0.9419	0.9419	86
Nat.	0.9333	1.0000	0.9655	14
Pla.	0.9000	0.9474	0.9231	76
Tre.	0.9667	0.9667	0.9667	60
Tri.	0.8462	0.9167	0.8800	12
War.	0.9452	0.9079	0.9262	76
accuracy			0.9332	419
macro avg	0.9140	0.9298	0.9213	419
weighted avg	0.9344	0.9332	0.9334	419

□



Two thin orange lines intersect on the left side of the slide. One line is horizontal, and the other is diagonal, crossing it.

Q&A

SOURCES

- Mesh Graphormer: <https://arxiv.org/pdf/2104.00272>
- Convolutional Mesh Regression for Single-Image Human Shape Reconstruction: <https://arxiv.org/pdf/1905.03244>
- [Yoga-Pose-Classification-and-Skeletonization²¹](#)