# **DSCI-6011**

# **FINAL PROJECT**

# **A blue logo with text and a book Description automatically generated**

# **Human Pose Estimation**

**Team Members**

Guna Jaswanth Reddy Maduri

Phanideepak Gaddam

Prudhvi Karthik Dhanekula

Mahesh Manchala

**Abstract**

Human pose estimation is a fundamental task in computer vision, with applications ranging from human-computer interaction to activity recognition and surveillance. In this report, we present our approach to human pose estimation using a machine learning model trained on a dataset comprising 5081 images annotated with landmarks describing human poses. We discuss our data collection process, data augmentation techniques, preprocessing steps, and the development of our model using the PyTorch framework. Our goal is to encapsulate the complex patterns of human movement into a mathematical model that can predict poses with a level of precision that is as close to reality as possible. Through meticulous curation of the dataset and careful optimization of our model, we aim to achieve high accuracy in pose estimation, paving the way for applications in various domains such as augmented reality, sports analysis, and healthcare.

**1. Introduction:**

In the world of computer vision, pose estimation is like teaching computers to understand how humans are positioned and moving in images or videos. It's super important for things like making cool augmented reality experiences or letting computers recognize hand gestures.

We're working on a project that uses machine learning to train a computer to figure out human poses. We're using a tool called MediaPipe made by Google, which is great because it's fast and accurate, even on everyday devices like your phone or laptop. Our goal is to make a model that can learn from a bunch of example images well, and then be able to figure out poses accurately even in pictures it's never seen before. If we can pull it off, this model could be used in all sorts of fun and useful apps, making technology feel more interactive and natural.

**2. Dataset:**

Our dataset comprises 5081 images, each annotated with landmarks describing human poses. These landmarks serve as strategic points on a human figure, defining posture and movement. To ensure standardized features across the dataset, we utilized the MediaPipe library to extract these key points from images.

**Data Partitioning:**

Following standard practice, we partitioned the dataset to facilitate training and evaluation of our machine learning model. We allocated 2500 images to the training set and reserved the remaining 1581 images for testing. This split is crucial for assessing the model's ability to

generalize beyond the data it was trained on.

**Data Augmentation:**

Data augmentation plays a pivotal role in enhancing the model's robustness. We implemented random horizontal flipping across the image set. This technique simulates the variability inherent in real-world scenarios, where the orientation of subjects can be unpredictable. Such augmentation helps prevent overfitting, ensuring that our model remains as versatile as possible.

**Preprocessing:**

In the preprocessing phase, every image was resized to a uniform dimension of 256x256 pixels. This standardization is necessary to maintain consistency in input data, which streamlines the computational process. Following resizing, the images were converted to tensor format to be compatible with the PyTorch framework, which was employed to develop the model.

**Normalization:**

Normalization of the dataset is a critical step in the data preprocessing pipeline. By calculating the mean and standard deviation of the images in the dataset, we normalized the image tensors. This process adjusts the pixel values so that they have a mean of zero and a standard deviation of one. Normalization significantly improves the convergence time of a neural network by scaling the inputs to a range where the network's activation functions can produce the most diverse derivatives, leading to more effective learning during the training phase.

Our dataset was meticulously curated to train a neural network for pose estimation. From data collection to normalization, each step was executed with the intention of developing a model that not only learns with high efficiency but also performs with admirable accuracy when exposed to new, unseen images. The final objective is to encapsulate the complex patterns of human movement into a mathematical model that can predict poses with a level of precision that is as close to reality as possible.

**3. Transfer Learning:**

In the realm of deep learning, transfer learning has emerged as a revolutionary technique, particularly in tasks where data is a luxury not easily afforded. It involves leveraging a pre-trained model — a model trained on a large benchmark dataset — and fine-tuning it to adapt to the specific nuances of a new, often smaller dataset. This method is predicated on the understanding that features learned by models in the initial layers are often generalizable to new tasks.

**4. RESNET Model Architecture**

A typical transfer learning approach would involve initializing a model with the architecture of a well-known pre-trained network like ResNet. These architectures have proven to be quite successful in capturing complex features from massive datasets like ImageNet. The output layer of such a model is usually replaced with a new layer tailored to the specific number of classes or predictions required for the new task — in this case, the coordinates of the pose landmarks.

**Base Network: ResNet-18**

* The base network used in this model is ResNet-18, a popular convolutional neural network architecture known for its effectiveness in various computer vision tasks.
* ResNet-18 consists of a series of convolutional layers followed by residual blocks, which allow for easier training of very deep networks.
* The ResNet-18 architecture comprises several layers, including convolutional layers, batch normalization layers, ReLU activation functions, max-pooling layers, and residual blocks.

**Customization:**

* The fully connected (fc) layer at the end of the ResNet-18 architecture is replaced with a new linear layer (nn.Linear) to adapt the network for the specific task of pose estimation.
* The output layer has num\_keypoints \* 2 output units, where num\_keypoints represents the number of keypoints to be detected in the pose estimation task. Since each keypoint has an (x, y) coordinate, we multiply num\_keypoints by 2 to get the total number of output units.
* The output of this model is a set of predicted coordinates (x, y) for each keypoint detected in the input image.

**Input and Output:**

* The input to the model (x) is an image tensor representing the input image.
* The output of the model is the predicted coordinates of keypoints in the input image.

The model utilizes the Mean Squared Error (MSE) as its loss function or objective function. This choice is quite common in regression tasks where the goal is to minimize the average of the squares of the differences between predicted and actual values.

**Experiments and Results:**

In the realm of traditional transfer learning, our experiments primarily focused on training the ResNet model from scratch. However, an essential component of effectively training neural networks lies in hyperparameter tuning. To address this, we leveraged Ray Tune to systematically explore the hyperparameter space, specifically targeting the optimal learning rate and weight decay for the Adam optimizer.

The learning rate plays a crucial role in determining the magnitude of parameter updates during optimization, while weight decay aids in regularization by penalizing large weights, thereby preventing overfitting. Through Ray Tune, we meticulously varied these parameters and evaluated the model's performance based on the validation loss.

This scheduler dynamically adjusted the learning rate by reducing it by a factor of 0.001 every 5 epochs. This strategy ensured that the model smoothly approached the minima of the loss function without overshooting. The best-performing model underwent evaluation on the test set to quantify its prediction accuracy using the Mean Squared Error (MSE) metric.

Overall, our approach encompassed a systematic exploration of hyperparameters, complemented by a dynamic learning rate decay policy, culminating in the selection of a model configuration with superior performance for subsequent evaluation on the test set.

**Constants:**

A close-up of a computer screen

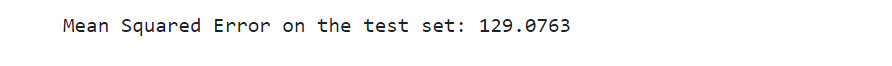
Description automatically generated

**Batch Loss:**

A screenshot of a computer

Description automatically generated

**MSE Loss:**



**Output:**

A person and person ice skating

Description automatically generated**A group of men playing football

Description automatically generated**

**5. Pose Estimator Model Architecture:**

The PoseEstimator model architecture comprises a series of convolutional and fully connected layers tailored for pose estimation tasks. Here's a breakdown of its architecture:

**Input Layer:**

Type: Convolutional layer (Conv2d) - Receives input images with RGB channels

Channels: 3 (corresponding to RGB) Kernel size: 3x3

Padding: 1 Output channels: 16 Output size: Same as input size

**Hidden Layer 1:**

Type: Convolutional layer (Conv2d) - Extracts features from the input images

Channels: 16 (input from previous layer) Kernel size: 3x3

Padding: 1 Output channels: 32 Output size: Half of the input size (due to max pooling)

**Hidden Layer 2:**

Type: Fully connected layer (Linear) - Processes the flattened feature map

Input size: 32 \* 64 \* 64 (computed from previous layer) Output size: 512

**Output Layer:**

Type: Fully connected layer (Linear) - Produces the final pose estimation output

Input size: 512 (output from previous layer)

Output size: num\_keypoints \* 2 (where num\_keypoints corresponds to the number of landmarks)

The model utilizes Rectified Linear Unit (ReLU) activation functions after each layer to introduce non-linearity and enhance learning capability.

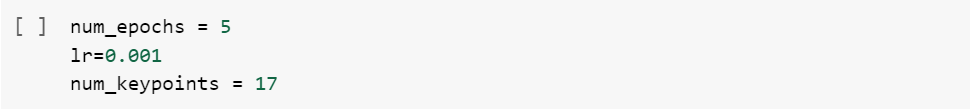
The objective function employed for training is the Mean Squared Error (MSE) Loss, which calculates the average squared difference between the estimated pose values and the ground truth. This loss function encourages the model to minimize the discrepancy between predicted and actual pose coordinates, thereby enhancing prediction accuracy.

**ii) Experiments and Results:**

The experiments with the PoseEstimator model revolved around finely tuning hyperparameters, leveraging Ray Tune's systematic approach to optimize the learning rate and weight decay. This meticulous tuning aimed to minimize validation loss, ensuring the model's effectiveness in predicting human poses accurately. Employing a learning rate decay policy with a StepLR scheduler played a crucial role in refining model weights as it converged, effectively preventing oscillations around the minima of the loss function. By systematically adjusting the learning rate every few epochs, the model could smoothly converge towards an optimal solution without overshooting or getting stuck in local minima.

Model selection was intricately tied to empirical results obtained from the experiments. The iteration exhibiting the lowest validation loss was identified as the optimal model configuration. Subsequently, this selected model underwent rigorous evaluation on a separate test set, where its performance was quantitatively assessed using Mean Squared Error (MSE). This evaluation process provided a robust measure of the model's predictive precision, ensuring that it could accurately estimate human poses across diverse scenarios.

**Constants:**

****

**Epochs Loss:**

**A screenshot of a computer screen

Description automatically generated**

**MSE Loss:**

****

**Output:**

**A screenshot of a football game

Description automatically generatedA person holding a child

Description automatically generatedA person and person on ice

Description automatically generated**

**6. Conclusion:**

The comparison between the ResNet model and the PoseEstimator model sheds light on the critical role of model architecture in achieving high predictive accuracy. Despite both models being trained for the same task of human pose estimation, the ResNet model, with an MSE loss of 129, significantly outperforms the PoseEstimator model, which has an MSE loss of 29401. This substantial difference in MSE loss highlights the superior ability of the ResNet model to capture the intricate patterns inherent in human poses from images.

The superiority of the ResNet model can be attributed to its more complex architecture and its pre-training on large benchmark datasets like ImageNet. By leveraging deeper layers and a more sophisticated feature extraction process, ResNet can effectively learn and represent the nuanced relationships between different body keypoints, resulting in more accurate pose estimations. In contrast, the simpler architecture of the PoseEstimator model may struggle to capture the complexity of human poses, leading to higher prediction errors. Therefore, in tasks where precision is paramount, such as human pose estimation for applications like augmented reality or gesture recognition, opting for established and more advanced architectures like ResNet proves to be a more effective approach.

In conclusion, the comparison between the ResNet and PoseEstimator models highlights the critical role of architectural complexity in achieving high performance in machine learning tasks. The ResNet model's superior performance, as evidenced by its significantly lower MSE loss, emphasizes the value of leveraging established architectures with deep layers and intricate feature extraction mechanisms. By opting for advanced architectures like ResNet, practitioners can effectively address complex pattern recognition tasks like human pose estimation, achieving superior predictive accuracy and enabling a wide range of applications in areas such as augmented reality, gesture recognition, and motion analysis.

**7. References:**

1. Cao, Zhe, et al. "Realtime multi-person 2d pose estimation using part affinity fields." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
2. OpenAI. "OpenAI PyTorch Scholarships for Minority Students." OpenAI Blog, 12 Jan. 2023.
3. Dong, Xing, et al. "Distillpose: Distillation of pose priors for 3d human pose estimation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.
4. Canseco-Aguilar, Andrés, et al. "Artificial intelligence to support physical therapy in Parkinson’s disease: Systematic review and bibliometric analysis." JMIR Rehabilitation and Assistive Technologies. 2022.
5. Google Research. "MediaPipe: A Framework for Building Perception Pipelines." Google AI Blog, 10 June 2021.
6. Facebook AI. "Detectron2." GitHub Repository.