An Exploration of ResNet Fine-tuning for Fitness Pose Classification

Nick Walker
Department of Computer Science
Stanford University
nhwalk13@stanford.edu

Ori Spector
Department of Humanities and Sciences
Stanford University
orispec@stanford.edu

Abstract

In the rapidly growing domain of remote fitness, the need for accurate pose recognition systems is paramount. However, current pose classification models often fall short in realistic fitness scenarios. This study addresses this challenge by leveraging the InfiniteForm dataset, a rich synthetic resource consisting of 60,000 images covering 15 diverse fitness pose categories, with variations in lighting, camera angles, and occlusions. Our research introduces a novel methodology combining pose estimation techniques with deep learning models to accurately classify fitness exercises. Utilizing the ResNet-50 model in tandem with relevant body keypoints features, we accurately classified various fitness poses. We show that fine-tuning a pre-existing state of the art model with pose estimation features greatly increases correct prediction of correctly labeled fitness exercises compared to a baseline pre-trained (on ImageNet) ResNet-50 model. Lastly, we analyze the effect of adding additional fully connected layers and ReLU activation's to test if this increases our best models accuracy.

1. Introduction

In this project, we aim to classify images into one of 15 different fitness pose classes based on their respective pose categories. The input to our method is an RGB image, and the output is the pose category of the image, represented as an integer between 0 and 14.

This classification problem is important to solve well so that the many opportunities that remote fitness presents can be capitalized on. For example, remote fitness apps could leverage the recognition of fitness poses to then provide instruction to users. Beyond this, our model could be used as a part of a larger model that first recognizes what fitness activity is being performed, and then analyzes key points on users' bodies to provide suggestions on pose correction. Moreover, our use of images rather than video and a 2D-CNN architecture lends itself well to potential deployment on edge devices, relative to more bulky 3D-CNN counterparts.

To accurately classify the various poses we employ the ResNet-50 architecture, which we fine-tune with both linear and non-linear classification heads. ResNet-50 is a variant of the ResNet (Residual Network) architecture, a popular convolutional neural network (CNN) for image classification tasks. It was proposed by Kaiming He and colleagues in their 2015 paper titled *Deep Residual Learning for Image Recognition* [6]. The "50" in ResNet-50 denotes that the network has 50 layers deep, including convolutional and fully connected layers. ResNet architectures, including ResNet-50, introduced the innovative concept of "residual learning" to mitigate the problem of vanishing gradients, which often occur in deep neural networks. This was achieved through the introduction of "shortcut" connections that allow the gradients to be backpropagated to earlier layers more effectively. The residual learning framework thus allows the training of much deeper networks than was previously possible, and these deeper networks achieve better performance. ResNet is considered state of the art for image classification[1].

In the context of our research, the *pose classification* problem, ResNet-50 can provide several benefits. First, it allows us to leverage the powerful feature extraction capabilities of a deep CNN without the need for extensive data or computational resources for training from scratch. Second, the depth of ResNet-50 means it can learn complex pose features effectively. Lastly, by fine-tuning ResNet-50 on our specific task, we can adapt the powerful general features learned on ImageNet [5] to our specific pose classification task, potentially achieving high performance with less data and computational resources than training a model from scratch.

1.1. Baseline

For our research, we used one core baseline: pre-trained ResNet-50 (i.e. transfer learning without fine-tuning) with

[1] ResNet was the Winner of ILSVRC 2015 in image classification, detection, and localization, as well as Winner of MS COCO 2015 detection, and segmentation

the final fully connected layer being replaced by a single untrained, randomly initialized linear layer mapping the output into 15 dimensions (one for each pose class). This produced results similar to random guessing.

Equivalently, our baseline is the ResNet-50 model applied to our problem without fine-tuning. We leverage the concept of transfer learning, wherein we use a pre-trained ResNet-50 model, trained on the ImageNet dataset, as our starting point. These pre-trained weights provide us with a rich feature extractor which has learned to discern various features from images. In this case, we use the pre-trained model as a fixed feature extractor. The output of this model (prior to the classification layer) serves as a high-level representation of our images, which can then be used in conjunction with a simpler classifier (like a linear classifier). However, without any finetuning, the model can't properly utilize any of this through the linear classification head. This baseline provides us with a sense of how our model performs without any training in a situation akin to random guessing. Thus, this baseline is crucial for our project as we seek to create a model that learns how to classify poses (this baseline enables us to see that it really is learning something) and explore how to effectively fine-tune ResNet-50 with relevant features and data to increase the classifier's accuracy on our task of exercise pose classification.

2. Related Works

In the pursuit of developing a precise and reliable model for fitness pose classification, our work is built upon established approaches in the field, notably the ResNet-50 architecture [6], and innovates upon these through a combination of unique integration of keypoint data and standard fine-tuning processes utilizing linear and non-linear classification heads with techniques like dropout [13], allowing for enhanced accuracy in pose identification. The general technical approach for deep learning image classification is founded on the use of CNNs, which AlexNet [10] proved was state-of-the-art[2]. Addtionally, we build off work from *Yoga Pose Classification Using Deep Learning* by Shruti Kothari [9] whose project seeks to develop an AI system for accurate, real-time yoga pose classification and instruction. He shows the potential effectiveness of keypoints as feature for Yoga, which we show generalizes to more general fitness pose classification.

2.1. OpenPose

OpenPose is a real-time system for joint multi-person 2D pose estimation, developed by Cao et al [2]. This method uses a novel architecture, called Part Affinity Fields (PAFs), to learn to associate body parts with individuals in the image. OpenPose can estimate the pose of multiple people in an image or video, making it useful for a wide range of applications, such as activity recognition, animation, gaming, and augmented reality.

OpenPose works by first predicting a set of 2D confidence maps (also called heatmaps) for each body part (e.g., left elbow, right knee) and a set of 2D vector fields (the Part Affinity Fields) which encode the degree of association between parts. These outputs are produced by a CNN architecture and then processed by greedy inference to obtain the final pose estimation for all people in the image. OpenPose shows that keypoints are a useful feature in pose estimation. Thus, we use this same intuition when adding the keypoints feature to our image classification model.

2.2. Recurrent Pose Classification

In a paper by Stanford students Chen et al. [3], they show the possibility for RNNs to accurately predict poses. The paper explores the possibility of modeling pose estimation as a sequence task using a convolutional network linked to a recurrent one. The authors tested this hypothesis and found that the CNN-RNN performed worse than the CNN. They also found that adding attention did not improve the result of the CNN-RNN. Further work is needed to expand upon specific reasons why the model performed worse than the baseline with additional quantitative results to test their hypothesis. Overall, the paper provides insights into the challenges and potential solutions for human pose estimation using deep learning techniques.

2.3. 3D Pose Classification

In research conducted by Mahendran et al . [11] they propose a novel approach to 3D pose estimation from a single image using a CNN regression framework. The authors argue that the 3D pose space is continuous, and therefore propose to solve the pose estimation problem in a regression framework rather than discretizing the pose space into bins and solving a classification problem.

The proposed approach uses a suitable representation, data augmentation, and loss function that captures the geometry of the pose space. The authors evaluate their approach on PASCAL3D+ and show that it achieves competitive performance compared to state-of-the-art methods.

While their paper focuses on 3D pose estimation, it provides useful context for our project on 2D image pose classification by highlighting some of the existing approaches in this area, especially regarding CNN-based methods that predict 2D keypoints from an image to solve the pose classification issue.

3. Methods

Our approach to fitness pose classification involves a series of models, all of which are based on ResNet-50. We

[2]CNN that was winner of ImageNet

employ transfer learning and fine-tuning techniques in our models. In particular, we experiment with the incorporation of pose keypoint features to potentially increase the performance of our pose classification, linear and non-linear classification heads of varying sizes (1 linear layer, 2 linear layers with ReLU and dropout between, and 3 linear layers with ReLU and dropout between), and freezing all of ResNet-50's layers vs. unfreezing the last layer of ResNet-50.

Using keypoints (obtained from the metadata from our dataset) to fine-tune ResNet-50, provides a feature that encapsulates important structural information about the objects in the images. In the case of human pose estimation, keypoints can represent different joints in the body, capturing the pose of the person in the image. By fine-tuning ResNet-50 on these keypoints, we essentially guide the learning process of the model towards recognizing these important structural features. This is particularly beneficial in tasks where the pose or arrangement of objects in the image is important. For example, in our task where we need to classify different types of fitness pose activities (pushups, squats, etc.), the relative position for keypoints of the person is a crucial piece of information. These keypoints are integrated into our model by being concatenated to the input to the first linear layer in the classification head, which also consists of the output of the second to last layer of ResNet-50 (we overwrote the last layer with our linear layer, but we refer to this second to last layer as the last layer of ResNet-50 because it essentially is for our purposes).

We hypothesize that integrating keypoints in the classification head, as described above, on top of the features learned from the inputted RGB images will allow the model to leverage structural information and yield more accurate results for our pose classification task. As for the other changes, we hypothesize that unfreezing the last layer will be beneficial by allowing our model to be more robust at training time. Finally, we hypothesize that the 3 linear layers with ReLU and dropout between will achieve the best accuracy since it is a robust non-linear classification head that has more parameters and more capacity to learn complex relationships in the data, with dropout preventing overfitting.

3.1. Methods

Baseline: Pre-Trained ResNet-50 For the baseline, we add a linear classification head to the pre-trained ResNet-50, overwriting the last fully-connected layer of ResNet-50. The linear classification head consists of 1 linear layer that takes in input from ResNet-50 and maps it down to a 15-dimensional space so that class scores can be computed. This linear layer is not trained and is randomly initialized. This baseline is the only "model" that was not run for 10 epochs, as the rest of the models were all trained with 10 epochs.

Model 1: ResNet (Frozen) + Linear Layer: In the first model, we freeze the weights of the pre-trained ResNet-50 and add a fully-connected linear layer as a linear classification head that is trained on our dataset. This effectively turns ResNet-50 into a fixed feature extractor, where the final linear layer is used to learn how these extracted features segment our specific classes. This is the same architecture as our baseline, except we allow for 10 epochs of training.

Model 2: ResNet (Unfrozen Final Layer) + Linear Layer: In the second model, we again freeze the weights of the pre-trained ResNet-50 model, but make exception for the final fully-connected layer of ResNet-50, which we unfreeze. We apply the same linear classification head as in the previous model, with the only change being the unfreezing of the final layer. This could allow for more effective fine-tuning of ResNet-50 by making our model's training more robust and tailored to our specific task.

Model 3: ResNet (Frozen) + Keypoints + Linear Layer: In the third model, we incorporate pose keypoint features into our model. We use the pre-trained ResNet-50 as a fixed feature extractor for the images, and combine these image features with the keypoints information in the linear layer of the linear classification head. We propose that the additional information from the pose keypoints may provide useful context for the classification task and improve performance. Explicitly, this builds on the same architecture as model 1, but with the addition of the keypoints feature.

Model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer: In the fourth model, we allow the weights in the final layer of ResNet-50 to be trainable while keeping the rest of ResNet-50 frozen, similar to model 2. We also concatenate the keypoints with ResNet-50's extracted features as an input to our still single-layer linear classification head, similar to model 3. This potentially will allow our model fine-tuning to be more robust for our specific task and also take advantage of the additional pose keypoints information. This model builds on model 2, but with the addition of the keypoints feature.

Model 5: ResNet (Unfrozen Final Layer) + Keypoints + 2 Linear Layers + ReLU/Dropout: In the fifth model, we build on top of model 4 by utilizing a non-linear classification head. This model also unfreezes ResNet-50's final layer and adds in keypoints. However, rather than having a single linear layer as a linear classification head, this model has a non-linear classification head that begins with a linear layer mapping the concatenated keypoints and ResNet-50 output to a 512-dimensional hidden layer. This is then pushed through a ReLU non-linearity and dropout with p=0.5 is applied. Finally, a linear layer maps the 512-dimensional hidden dimension to a 15-dimensional output space. We propose that this shift to a non-linear classifi-

cation head will allow our model to learn more complex relationships in the data and will avoid overfitting by using dropout as a regularizer.

Model 6: ResNet (Unfrozen Final Layer) + Keypoints + 3 Linear Layers + ReLU/Dropout: In the sixth model, we build on model 5 by utilizing a non-linear classification head, except we now add yet another linear layer with non-linearities, while everything else remains consistent. This model begins with a linear classification head mapping the concatenated keypoints and ResNet-50 output to a 512-dimensional hidden layer. This is then pushed through a ReLU non-linearity and then dropout with p=0.5. Following this, the output is fed to a second linear layer which maps the 512-dimensional space down to a 256-dimensional hidden layer. Once again, ReLU and then dropout with p=0.5 are applied before a final linear layer maps the 256-dimensional hidden layer to a final 15-dimensional output layer.

3.2. Data pipeline

PoseDataset class: We define a custom PyTorch Dataset, PoseDataset, to load the fitness pose images and their associated labels and keypoints. The keypoints are included as an additional feature along with the image data. The purpose of this class is to handle the dataset of images and corresponding labels and keypoints. It's initialized with the directory containing the images, a dictionary of annotations corresponding to the images, and a series of image transformations (normalized to extract ImageNet features). In the __getitem__ method, for each image ID, it retrieves the image, the corresponding pose category as an integer, and the keypoints as a tensor.

3.3. Additional Shared Model Details

The architectures of our neural networks are based on the pre-trained ResNet-50. Using PyTorch, we load a pre-trained ResNet-50 model. This model is designed to receive images as input and transform them through a series of convolutional and pooling layers, learning to extract numerous features from the input images. Since we use the pre-trained model, it comes with weights that have been already trained on a large dataset (ImageNet), thus having already learned to extract a wide range of features from images.

The training and testing datasets are created using the PoseDataset class and the respective image directories. Dataloader objects are created for each of these datasets which will allow for batch loading of the data during training and evaluation. We then initialize an instance of our deep learning model.

For our loss function, we applied cross-entropy loss, as it is commonly used for multi-class classification problems and is more precise than alternatives like hinge loss. With respect to our optimizer, we use the Adam optimizer [8]. For our models, the optimizer updates only the parameters of the final fully connected layer of ResNet-50 (if it is unfrozen), as well as all other parameters in the classification heads.

For each epoch, the model is set to training mode. On each batch of images, targets, and associated keypoints (if relevant), the model's forward method is called, and the cross-entropy loss between the model's predictions and the true targets is calculated. This loss is then backpropagated through the model and the optimizer updates the model's parameters. The learning rate scheduler adjusts the learning rate (which starts at 0.001) at the end of every 7 epochs with gamma=0.1 as the decay rate (PyTorch's default). After each epoch of training, the model is evaluated on the test set and we keep track of our best model up to that point. The model's forward method is called with no gradients being calculated and the loss and accuracy on the test set are recorded. This process is repeated for the number of epochs specified (10). The end result is a model that is effective at classifying fitness poses, with the training process guided by both the image data and the additional pose keypoints.

4. Dataset and Features

We utilize the InfiniteForm dataset developed by Weitz et al [14], an open-source synthetic collection of 60,000 images, for our pose classification task. It comprises diverse fitness poses categorized into 15 unique categories, involving both single and multi-person scenes. This dataset is particularly suited to our task due to its inclusion of realistic variations in lighting, camera angles, and occlusions, which mirror real-world fitness scenarios. [3]

Notably, the InfiniteForm dataset exhibits minimal bias concerning body shape and skin tone, making it an excellent tool for developing unbiased fitness tracking applications. Each image in the dataset is accompanied by pixel-perfect labels for standard annotations like 2D keypoints, as well as more complex annotations such as depth and occlusions.

The dataset provides comprehensive annotations for each scene, including RGB images, semantic segmentations, instance segmentations, and 32-bit, unnormalized depth maps. For every avatar, there are additional labels and metadata, including 2D keypoints in standard COCO format, 3D keypoints, polygon segmentation in standard COCO format, bounding box in standard COCO format, and several other features. We utilized the labeled 2D keypoint data as an additional feature for our model, beyond the RGB images themselves.

Before training our model, we performed several preprocessing steps on the dataset. We split the dataset into a training set of 50,000 images and a test set of 10,000 images.

[3] In the paper Weitz et al show that there is minimal bias in their generated dataset.

For each image, we applied a series of transformations. The images were resized to a standard size of 224x224 pixels and then converted to tensors. Finally, we normalized the images with mean and standard deviation values of [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], respectively, to match the normalization parameters of the pre-trained ResNet-50 model we used.

To ensure the efficient use of computational resources, we grouped the images into batches of 64. This batching facilitates faster training times by enabling the simultaneous processing of multiple images.

(a) RBG image (b) RBG Image + Keypoints

Figure 1. Sample Data

5. Experiments, Results, & Discussion

In this section, we highlight the hyperparamters we used for our models, and the rationale behind the choices. Moreover, we present the metrics we used to assess our methods. These help form the results of the paper, which we present via various graphs and tables. Lastly, we discuss our results.

5.1. Hyperparameters

We utilized the Adam optimizer's default learning rate, which is 0.001. Adam is a popular choice for deep learning tasks due to its leveraging of momentum and an adaptive learning rate, as well as its ability to handle sparse gradients and noisy data, which is why it was selected [8]. Additionally, the batch size was set to 64, as it offers a good balance between memory usage and computational efficiency. Smaller batch sizes can provide more frequent updates, but they may also be noisier, while larger batches offer more stable but less frequent updates[4]. We used the StepLR learning rate scheduler to decrease the learning rate by a factor of 0.01 every seven epochs. This combination of learning rate and batch size has been shown to be effective in training CNNs to help the optimizer converge to the optimal solution more quickly and accurately [12]. We trained our model for 10 epochs because that balanced model learning and training time, which was important given the large amount of models we were training.

Moreover, We experimented with freezing and unfreezing of the final layer of ResNet-50 because it is common practice in transfer learning [4]. By unfreezing the last layer of ResNet50, we allowed the model to adapt more flexibly to the specific task at hand, while still benefiting from the pre-trained weights of the earlier layers which could already capture lower-level details that are useful for our specific task too.

Additionally, we used different model architectures such as a simple linear classification head and more complex non-linear classification heads which help to capture more complex relationships in the data[5]. For example, in our 2-linear-layer non-linear classification head, we introduced a linear layer followed by a ReLU activation function that was then followed by a dropout layer and another linear layer. The ReLU function introduces non-linearity into the model, enabling it to learn more complex mappings from the inputs [1]. The dropout layer with a rate of 0.5 is used to prevent overfitting by randomly setting half of the input units to 0 during training, which helps to prevent the model from relying too heavily on any single input or becoming too complex. The rate of 0.5 is commonly used and recommended in the literature, for example, in Geoffrey Hinton's work [13]. Dropout was introduced in models 5 and 6 (those that used non-linear classification heads) to prevent the overfitting that is often associated with more robust models.

5.2. Metrics

To evaluate the performance of both our method and the baselines we utilize loss and accuracy metrics. To compute loss we use cross-entropy loss or softmax.

$$L_{CE} = -\sum_{i=1}^{n} t_i * log(p_i), \text{ for n classes} \quad (1)$$

To compute the model's classification accuracy, we iterate through each batch and calculate the proportion of images where the predicted pose category matches the actual pose category. We sum these proportions and divide by the number of images to get the training accuracy for an epoch.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (2)$$

5.3. Results

We show the accuracy and loss for all are models in the tables 1 and 2. Moreover, for our best performing model [Model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer], we plot loss and accuracy over time (10 epochs) in figures 2a and 2b. Additionally, we provide a confusion matrix for this model to show class by class predictions in figure 3. Furthermore, again for model 4,

[4] Additional research adds insight on large batch sizes [7]

[5] We use 512 and 256 units for fully connected layers since they are both powers of two (common practice) and reflect the balance between computational efficiency and task complexity

in figures 4a, 5a, 4b and 5b we display saliency maps to visualize which parts of an input image contribute most to the model's output. The saliency map was created by computing the gradient of the output with respect to the input image. The larger the absolute value of the gradient at a certain pixel, the more "saliency" or importance the model assigns to that pixel. We present two examples of saliency maps, one in which our classifier is correct and another where the pose category was misclassified, alongside the original images they were produced from.

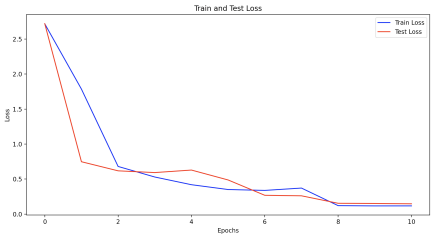
Table 1. Model Loss Comparison

Model	Train	Test
Baseline (Pre-Trained Res-Net50)	2.7185	2.7207
Model 1: ResNet (Frozen) + Linear Layer	2.0516	2.7342
Model 2: ResNet (Unfrozen Final Layer) + Linear Layer	0.0110	2.5679
Model 3: ResNet (Frozen) + Keypoints + Linear Layer	0.1155	1.1266
Model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer	0.1165	**0.1462**
Model 5: ResNet (Unfrozen Final Layer) + Keypoints + 2 Linear Layers + ReLU/Dropout	0.8509	0.9214
Model 6: ResNet (Unfrozen Final Layer) + Keypoints + 3 Linear Layers + ReLU/Dropout	1.1333	0.8190

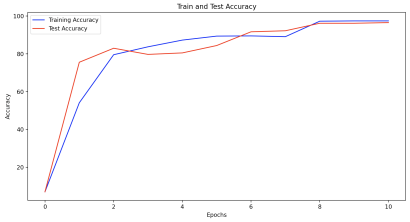
Table 2. Model Accuracy Comparison

Model	Train	Test
Baseline (Pre-Trained Res-Net50)	6.98%	6.90%
Model 1: ResNet (Frozen) + Linear Layer	35.91%	19.30%
Model 2: ResNet (Unfrozen Final Layer) + Linear Layer	99.80%	31.98%
Model 3: ResNet (Frozen) + Keypoints + Linear Layer	97.58%	59.00%
Model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer	97.45%	**96.55%**
Model 5: ResNet (Unfrozen Final Layer) + Keypoints + 2 Linear Layers + ReLU/Dropout	66.90%	64.50%
Model 6: ResNet (Unfrozen Final Layer) + Keypoints + 3 Linear Layers + ReLU/Dropout	58.80%	79.31%

We have bolded model 4 to show that it is our best model in terms of test accuracy and loss.



(a) Loss over Epochs



(b) Accuracy over Epochs

Figure 2. 1FC + keypoints

This plot highlights that the models accuracy increases and the loss decreases over training time. This served as a good indication that our model was predicting well during training and showed improvements as compared to previous models.

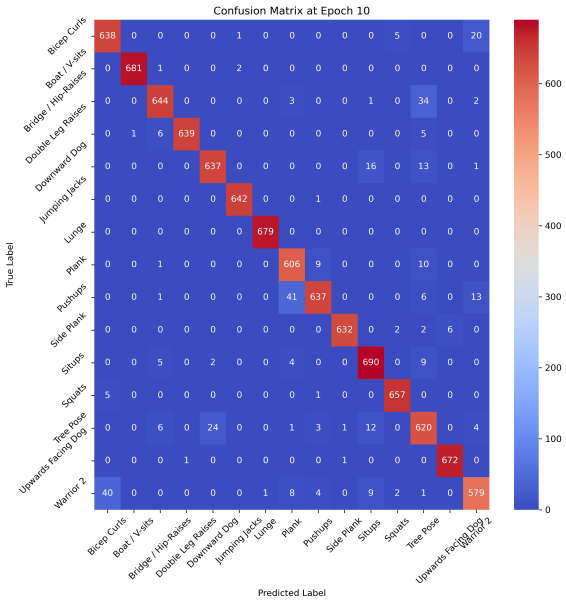
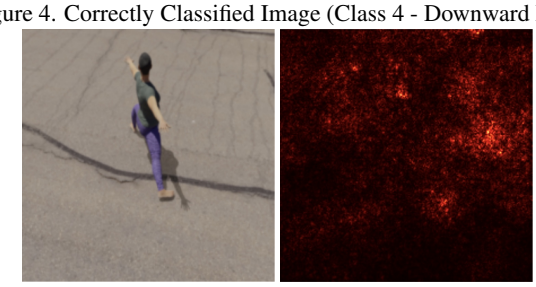


Figure 3. Confusion matrix

Our confusion matrix displays a class by class breakdown of correct and incorrect predictions as compared to the ground truth labels. This shows that warrior 2 and bicep curls caused the most confusion for our network, consequently causing the most errors.

(a) RGB Image (b) Saliency Map

Figure 4. Correctly Classified Image (Class 4 - Downward Dog)



(a) RGB Image (b) Saliency Map

Figure 5. Incorrectly Classified Image (Correct Warrior 2, Predicted Lunge)

In figure 4b we see that the saliency map correctly outlines the person in the downwards dog position as there is a

clear V shape on the left side of the map. Additionally, we see that the area with the most bright read is on the shadow of the figure, potentially highlighting the fact that our model learned from these as well. In figure 5b, our model gets distracted by the patterns on the ground in the background of the image. We can see this by noticing that the most red areas are following the lined patterns on the ground. This is what likely caused our model to inccorect predict the pose.

5.4. Experiments and Discussion

We noticed that our baseline model utilizing the pre-trained ResNet-50 weights performed roughly similar to random guessing. This was because the linear layer was randomly initialized and no training was performed, so the output of the layer was essentially nothing more than random matrix multiplications.

As for our models, we had three key findings: 1) Adding in keypoints did indeed greatly increase the performance of our models, suggesting that the spatial information they provide was significantly helpful for our model when determing pose class, 2) Unfreezing the last linear layer of ResNet-50 also significantly helped our model performance, indicating that this delicate unfreezing of a single layer had tremendous impacts on the model's ability to adapt to our specific task, and 3) Despite what we believed, adding non-linear classification heads did not improve performance and were outperformed by a linear classification head, at least across the 10 epochs that we ran.

For key finding 1, we see this evidenced by the increase in performance from Model 1 (19.30% Test Accuracy) to Model 3 (59.00% Test Accuracy) and by the increase in performance from Model 2 (31.98% Test Accuracy) to Model 4 (96.55% Test Accuracy). These models were both the same architecturally, only differing by the addition of keypoints. The substantial difference in test accuracy shows the usefulness of keypoints in a fitness pose estimation setting.

For key finding 2, we see this evidenced by the increase in performance from Model 1 (19.30% Test Accuracy) to Model 2 (31.98% Test Accuracy) and by the increase in performance from Model 3 (59.00% Test Accuracy) to Model 4 (96.55% Test Accuracy). Again, both of these increases are substantial. We can attribute this change due to the fact that our model was more robust and was able to take advantage of adjusting more weights during fine-tuning, allowing earlier frozen layers to capture general low-level (like edges, colors, shapes) details while the last unfrozen layer was able to be fine-tuned to the specific task at hand. In other words, the last layer, when unfrozen, can specialize on the specifics of our task, thus improving accuracy. Since ResNet-50 is good for image classification, but not specifically for fitness poses (with neither just one nor multiple people), this helped significantly with allowing the pre-trained network's weights to be properly applied and leveraged. This shows the importance of adjusting pre-trained model weights when fine-tuning in the fitness pose estimation setting.

For key finding 3, we see this evidenced by model 4's significantly better performance (96.55% Test Accuracy) compared to model 5 (64.50% Test Accuracy) and model 6 (79.31% Test Accuracy). Model 4 is the same as model 5 and 6, except for the fact that it uses a linear classification head while models 5 and 6 use non-linear classification heads. Despite what we believed would occur, model 4 outperformed the models that were more robust and complex. This could partially be due to it being less likely to overfit (we saw some overfitting in model 5 and in earlier models where training accuracy was better than test accuracy) or due to linear classification heads simply performing better for the fitness pose estimation task. However, it is also likely that models 5 and 6, being as complex as they were, weren't able to fully tune their many additional parameters in only 10 epochs. The relatively simpler model 4 could have been able to achieve closer to its maximum potential within 10 epochs since it has less parameters that therefore interact with each other in less complex ways that can be learned in a shorter period of "time" (epochs).

Taking a step back and looking at our results beyond these three key findings, model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer performed best by far, achieving a 96.55% test accuracy. It was also apparent that our model drastically overfit our data for models 1,2, and 3. This is clear because the accuracy was much higher for train than test. Also, the loss was much lower for train than test. Models 5 and 6 model combatted this overfitting issue by adding dropout. Notably, the 4th method does not use dropout, but does not demonstrate the same level of severity with the overfitting issue (it has only a slightly higher training accuracy at 97.45% than its 96.55% test accuracy), likely because the performance is so high that this discrepancy can't be large and it's capable of learning effective, generalizable features.

6. Conclusion

This paper demonstrates the effectiveness of integrating body keypoints features, unfreezing fully connected layers, and linear classification heads when fine-tuning ResNet-50 on the fitness pose classification task. Our exploration began with a baseline model leveraging pre-trained ResNet-50 weights, which showed a performance no better than random guessing due to the lack of initial training on our dataset. We were able to far surpass this baseline with every additional model we trained, demonstrating the learning capabilities of neural networks on the fitness pose classification task.

Significant performance improvement was observed when keypoints were incorporated into the model, suggest-

ing that spatial information provided by these keypoints is crucial for the task of fitness pose classification. Moreover, unfreezing the last linear layer of the ResNet-50 model resulted in a substantial increase in accuracy, underlining the benefits of selective layer unfreezing when fine-tuning models for specific tasks. Notably, Model 4: ResNet (Unfrozen Final Layer) + Keypoints + Linear Layer performed the best.

Contrary to our expectations, the addition of non-linear classification heads did not enhance the performance. Instead, a simple linear classification head outperformed the more complex non-linear models, possibly due to it being less prone to overfitting, or perhaps because the more complex models could not fully optimize their parameters within the limited training epochs we employed.

Our research illustrates that it is possible to significantly enhance fitness pose classification by strategically adapting pre-existing models and by effectively integrating spatial data from pose keypoints. However, this study also raises intriguing questions about the most effective model complexity and the benefits of nonlinear classifiers in such tasks, which warrants further investigation.

Given more time and resources, future studies could aim to experiment with larger, more diverse datasets and explore more complex model architectures trained on more epochs. This could also include a deeper analysis of the impact of non-linear activation functions in the classification layer and the effects of a more comprehensive training regime, potentially uncovering further areas for improvement in this essential domain. Additionally, more data could be fed to the model, like segmentation masks. Lastly, fine-tuning hyperparameters, such as experimenting with unfreezing various layers or combinations of layers in ResNet-50, and adjusting dropout rates, could present intriguing avenues for exploration in future research.

7. Contributions & Acknowledgements

We both evenly distributed the work for fine-tuning and training our various models via pair programming. Additionally, we worked together to complete our proposal, milestone, and research paper, with both team members editing and writing portions of each section. A central piece of our research could not be made possible without ResNet and the research from He et al. [6]. Thanks to Weitz et al. [14] for the free and comprehensive dataset without this we could not train our image classifier. Pytorch documentation was useful for loading the ResNet-50 model which can be found here.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. 5

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 2

[3] Matthew Chen and Melvin Low. Recurrent human pose estimation. 2016. 2

[4] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. *CoRR*, abs/1902.10547, 2019. 5

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 2, 8

[7] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima, 2017. 5

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4, 5

[9] Shruti Kothari. Yoga pose classification using deep learning, 2020. 2

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 2

[11] Siddharth Mahendran, Haider Ali, and Rene Vidal. 3d pose regression using convolutional neural networks, 2017. 2

[12] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161:11–19, aug 2017. 5

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 2, 5

[14] Andrew Weitz, Lina Colucci, Sidney Primas, and Brinnae Bent. Infiniteform: A synthetic, minimal bias dataset for fitness applications. *CoRR*, abs/2110.01330, 2021. 4, 8