[1]

Multiclass Yoga Pose Recognition

Joan Company Company & Adrià Cortés Cugat

Abstract: This project presents a deep learning-based yoga pose classification system using image data and convolutional neural networks. The objective is to develop an accurate and computationally efficient model capable of recognizing multiple yoga postures from still images. We evaluate several convolutional architectures, including lightweight models such as ConvNeXt-Tiny and EfficientNet-B2, to assess trade-offs between accuracy, computational cost, and training behavior. To address class imbalance in the dataset, we employ a class-weighted sampling strategy during training, ensuring equitable representation of all classes in each batch. The models are trained with transfer learning and enhanced with data augmentation to improve generalization. Our experimental pipeline includes hyperparameter tuning and comparative analysis across models. Results show strong classification performance, with detailed examination of confusion patterns and model limitations. This work demonstrates the viability of deploying yoga pose recognition systems in real-world fitness and wellness applications, especially where responsive and resource-efficient models are needed.

I. Introduction

The growing demand for intelligent systems in wellness and human activity monitoring has fueled research into the automatic recognition of body postures, particularly in the domain of yoga practice. Accurately classifying yoga poses through computer vision presents specific challenges, including pose variability, subtle visual differences between classes, and limited labeled data. Overcoming these challenges has practical relevance for applications such as personalized fitness coaching, injury prevention, and interactive feedback systems for remote training.

This project addresses the task of multiclass yoga pose recognition from images, using convolutional neural networks (CNNs) as the core modeling approach. Our objective is to develop an end-to-end image classification pipeline that maximizes accuracy while maintaining computational efficiency, which is a crucial consideration for real-time or embedded deployment scenarios. The dataset consists of 2,759 images distributed across 47 distinct pose classes, resulting in a highly imbalanced and sparse data distribution that poses a significant challenge for training robust models.

To meet these objectives, we investigate a range of CNN architectures, from lightweight custom models to fine-tuned pretrained networks. We also integrate data augmentation, stratified dataset splitting, and class-balanced sampling to address issues of class imbalance and improve generalization. The outcome is a robust, well-validated system that demonstrates the feasibility of applying deep learning to yoga pose recognition with limited data and constrained resources.

II. State Of The Art

Yoga pose recognition is an active area of research within computer vision due to its applicability in wellness technologies, fitness guidance, and human-computer interaction. The task typically involves recognizing static or dynamic postures from visual input, and it presents notable challenges such as subtle inter-class differences, varying body orientations, and visual noise from backgrounds or clothing.

Earlier approaches relied on handcrafted features combined with traditional classifiers (e.g., HOG + SVM), but these methods were limited in their ability to generalize to unseen poses and real-world conditions. The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has significantly advanced the field by enabling automatic extraction of hierarchical visual features [1]. CNNs have become the de facto standard for image-based posture recognition tasks due to their capacity to learn robust spatial patterns.

Transfer learning has emerged as a powerful strategy to compensate for limited task-specific data. Pretrained models like MobileNetV2 [2], EfficientNet [3], and Vision Transformers (ViTs) [4], initially trained on large datasets such as ImageNet, have been successfully adapted for yoga pose classification through fine-tuning. For example, ViTs have been reported to achieve accuracy levels above 92% on large-scale yoga datasets such as Yoga-82, demonstrating the potential of transformer-based architectures for fine-grained classification tasks [4].

Several public datasets have supported the development of this field:

- Yoga Posture Dataset (used in this project): 2,759 images across 47 classes, suitable for lightweight training and evaluation on consumer-grade hardware [5].
- Yoga-82: A large-scale dataset with over 28,000 images and 82 categories, offering a broader but more challenging classification task due to higher intra-class variability [4].

Some methodologies also explore the use of pose keypoint extraction and 2D skeleton-based representations, followed by classification with shallow neural networks. While these pipelines can enhance interpretability and offer benefits in certain applications like rehabilitation, they require integration

[1]Date on which the paper was submitted: 9/06/2025. F. A. Joan Company is with the University Pompeu Fabra, Barcelona, Spain. S. B. Adrià Cortés., was with University Pompeu Fabra, Barcelona, Spain. (correspondence F. A. e-mail: joan.company01@estudiant.uof.edu, S. B. e-mail: adria.cortes01@estudiant.upf.edu).

of multi-stage systems and are often more complex to deploy in purely image-based inference settings.

In our work, we focus exclusively on image-based recognition using CNNs. We evaluated custom shallow architectures alongside modern pretrained models such as ConvNeXt-Tiny [6] and EfficientNet-B2 [3]. These architectures offer a favorable balance between parameter efficiency and classification accuracy, particularly when combined with data augmentation and class-balanced sampling. ConvNeXt, in particular, retains the simplicity of CNNs while incorporating design principles from transformers, and has demonstrated strong performance on diverse image classification tasks [6].

Overall, the literature supports the effectiveness of CNN-based transfer learning pipelines for yoga pose classification, especially when adapted to the scale, structure, and constraints of specific datasets. Our project builds on this foundation, targeting efficient, scalable models for practical wellness-oriented applications.

III. Methodology

A. Data Analysis: Dataset and Preprocessing

This project uses the publicly available Yoga Posture Dataset [5], which contains 2,759 .png images distributed across 47 yoga pose classes. Each class is stored in its own folder, and images exhibit natural variability in lighting and backgrounds, introducing useful realism but also increasing classification difficulty.

Several key challenges characterize this dataset:

- A limited overall sample size relative to the number of classes.
- Severe class imbalance, with some poses represented by fewer than 20 images.
- High inter-class similarity, particularly among visually similar or transitional postures.

All images were resized to 224×224 pixels to comply with the input requirements of standard CNN architectures. We performed a stratified split of the dataset into:

- 70% training,
- 15% validation,
- 15% test, ensuring consistent class proportions across all splits.

To reduce overfitting and improve model generalization, we implemented a data augmentation pipeline using torchvision.transforms. Initially, we applied modest transformations such as horizontal flips and minor rotations. However, early training experiments revealed that these augmentations were not diverse enough to induce generalization across all models. In response, we exaggerated the augmentation strength, expanding the variability in training images.

The final augmentation pipeline included:

- Random horizontal flip (50% probability),
- Random rotation up to ±15°,
- Random resized crop with varied scale and aspect ratio,
- Color jittering (modifying brightness, contrast, saturation, and hue),
  Conversion to tensors and normalization using ImageNet statistics:
  mean = [0.485, 0.456, 0.406],
  std = [0.229, 0.224, 0.225].

These augmentations were applied only to the training set, while validation and test sets were resized and normalized without augmentation. After exaggerating the augmentations, we observed a consistent performance improvement across multiple models, demonstrating the positive impact of increased visual diversity.

To mitigate the effects of class imbalance, we implemented a class-weighted sampling strategy. We calculated inverse frequency-based weights for each class and used them to construct a WeightedRandomSampler in PyTorch. This ensured that each batch was representative of all classes without over-replicating underrepresented samples — a necessary consideration given the dataset's limited size and high imbalance risk.

All transformations and sampling procedures were verified visually. We also performed exploratory data analysis, including class distribution histograms and augmentation previews, to support preprocessing choices and ensure label consistency.

B. Model Design and Optimization

The modeling phase was structured in two stages: (1) Establishing a baseline using a custom CNN, and (2) Experimenting with transfer learning using pretrained architectures.

We first implemented a simple CNN from scratch to validate dataset integrity and confirm the feasibility of the classification task. This baseline helped define a lower-bound performance and guided early debugging and augmentation effectiveness.

To improve accuracy and efficiency, we evaluated several pretrained models, including:

- ResNet18 and DenseNet121, for classic CNN baselines,
- EfficientNet-B2, known for compound scaling and parameter efficiency [3],
- ConvNeXt-Tiny, a modern CNN that incorporates transformer-inspired design while retaining full convolutional structure [6].

These models were fine-tuned by replacing their final classification layers with custom linear heads matching the 47 yoga classes. Training was performed using Adam optimizer, categorical cross-entropy loss, and early stopping on the validation F1-score. Dropout regularization was used in the custom heads to reduce overfitting.

In parallel, we tested a lightweight custom CNN derived from a previously tuned model for digit recognition (SVHN). It employed depthwise separable convolutions inspired by MobileNet [2] and served as a parameter-efficient alternative to larger backbones.

After comparing all tested architectures, EfficientNet-B2 and ConvNeXt-Tiny emerged as the top-performing models in terms of accuracy, generalization, and efficiency. To further refine their performance, we conducted targeted hyperparameter tuning on these two models. The goal was to optimize training stability and convergence dynamics, and to evaluate whether careful tuning could yield additional performance improvements beyond architecture selection alone.

C. Optimization Strategy

- Batch size: 32
- Learning rates: searched in [1e-4, 1e-3]
- Early stopping: patience of 10 epochs
- Scheduler: CosineAnnealingLR in selected experiments
- Loss: Standard cross-entropy loss with class-weighted sampler applied through the DataLoader

The final selected architectures, EfficientNet-B2 and ConvNeXt-Tiny, were chosen based on their trade-off between accuracy and computational efficiency, as well as their capacity to generalize under data-scarce conditions.

This methodological framework enabled a thorough exploration of model performance under realistic constraints of data sparsity, imbalance, and deployment suitability. The results from both custom and pretrained CNNs provided insight into architecture efficiency, sampling strategies, and augmentation impact, forming a solid foundation for subsequent analysis and discussion.

IV. EXPERIMENTATION

The experimental phase of the project was designed to systematically evaluate various convolutional neural network (CNN) architectures for the task of multiclass yoga pose classification. Each set of experiments explored a different modeling or optimization strategy, with the objective of understanding the trade-offs between model capacity, generalization ability, training stability, and inference efficiency. All training and testing were conducted in PyTorch [7] on Google Colab using Tesla T4 GPUs.

A. Baseline and Custom Architectures

The initial baseline model was a simple CNN composed of three convolutional layers followed by a flattening and a fully connected classifier. While it achieved over 57.18% training accuracy, the test accuracy stagnated at 42.82%, showing a severe gap between training and validation curves. A confusion matrix revealed collapsed predictions toward dominant classes which could be a symptom of overfitting, poor feature extraction, and insufficient data augmentation.

To explore whether lightweight architectures could improve generalization while reducing complexity, we implemented a custom CNN using depthwise separable convolutions, inspired by MobileNet [2]. Despite a substantial parameter count of over 51 million, the model exhibited very low generalization capacity, achieving only 24.64% test accuracy and an F1-score of 0.24, with performance nearly identical across train, validation, and test sets, indicating underfitting.

We further experimented with a more compact architecture, LightweightNet, which included inverted bottlenecks, Batch Normalization, and Squeeze-and-Excitation blocks, reducing the parameter count to under 200k. However, this model also failed to generalize, achieving just 16.91% test accuracy and an F1-score of 0.13. These results confirmed that, given the fine-grained nature of the task and the limited training data, training from scratch (regardless of model size), was insufficient for achieving robust performance.

B. Transfer Learning of Lightweight CNNs from SVHN

To explore the trade-off between speed, parameter efficiency, and accuracy, we conducted experiments using lightweight CNN models originally trained on the SVHN digit classification dataset. These models, such as SmallMobileNet variants and ResSimDW hybrids, were optimized for 10-class digit recognition on 32×32 resolution images, and were characterized by low parameter counts and high inference speed.

Our motivation was not to outperform larger models, but rather to assess whether these compact architectures could deliver acceptable performance in real-time settings when transferred to a fine-grained pose classification task like ours. The aim was to empirically evaluate how source task/domain and model scale affect transferability, especially under strict latency constraints.

To do this experimentation, the yoga image data was resized accordingly for compatibility and two transfer strategies were explored:

C. Frozen Feature Extractors

In the first setting, all convolutional layers were frozen and only the classification head was trained. Input images were resized to 32×32 to match the resolution of the original SVHN models. Performance was generally poor:

- ResSimDW_BatchNorm achieved the best result with 47.58% test accuracy and an F1-score of 0.45.
- Most MobileNet variants underperformed, with test accuracies below 5% and F1-scores near zero (e.g., SmallMobileNetV2: 2.42%, F1: 0.01).
- Despite poor accuracy, these models were extremely fast, with inference times as low as 0.000058 sec/image.

These results illustrate the limitations of transferring from low-resolution, domain-specific tasks. Frozen feature extractors trained on digits failed to generalize to the complexity of human pose data, often collapsing predictions into dominant classes.

D. Partial Unfreezing

We then tested partial fine-tuning by unfreezing top convolutional blocks. This improved results moderately:

- ResSimDW_BatchNorm reached 57.49% test accuracy, F1: 0.57.
- Other ResSimDW variants showed smaller gains (e.g., ResSimDW: 39.61%, F1: 0.40).
- MobileNet-based models remained ineffective, with accuracies mostly under 5%.

Although partial unfreezing improved performance in certain cases, these lightweight models consistently lagged behind the larger ImageNet-pretrained baselines. Furthermore, the downsampled input resolution fundamentally limited their ability to capture fine-grained spatial and contextual cues necessary for distinguishing among 47 yoga poses.

In addition to underperformance, the downsampled input size severely constrained the spatial and contextual cues needed for pose differentiation, such as limb orientation and joint angles. As a result, even compact and fast models like these could not reliably capture the complexity of 47-class yoga classification.

These findings suggest that while SVHN-tuned models offer computational advantages (e.g., inference times below 0.0004 sec/image), naive cross-domain transfer from unrelated, low-resolution tasks is insufficient. Unlocking the potential of such architectures would require either higher-resolution inputs or large-scale task-specific pretraining.

E. Transfer Learning with Pretrained CNNs from ImageNet

After evaluating lightweight and SVHN-tuned architectures, we turned to transfer learning using pretrained CNNs from ImageNet, aiming to leverage strong visual representations learned from large-scale, high-resolution image data. This stage focused on systematically comparing models in terms of accuracy, generalization, parameter efficiency, and inference speed.

We tested a diverse range of architectures:

- Standard CNNs: ResNet18, ResNet34, ResNet50, Wide-ResNet50-2, ResNeXt50_32x4d
- DenseNet variants: DenseNet121, DenseNet169
- Lightweight CNNs: MobileNetV2, MobileNetV3-Large, ShuffleNetV2, SqueezeNet1.1
- EfficientNet family: EfficientNet-B0, B1, B2 [3]
- Modern CNNs: ConvNeXt-Tiny [6]

Each model's convolutional base was initialized with pretrained weights. We replaced the final classifier with a custom head consisting of global average pooling, dropout, fully connected layers, and a softmax output over 47 classes.

Among all models, ConvNeXt-Tiny delivered the best results with 89.37% test accuracy, F1-score of 0.89, and fast inference (0.000571 sec/image), making it the most promising candidate for deployment.

ResNet34 and ResNeXt50_32x4d also showed strong performance (F1-scores ≈ 0.87) with moderate size and inference time, offering a balance between accuracy and efficiency.

EfficientNet-B1 stood out for its compactness (6.5M parameters) while achieving 86.23% accuracy, making it a solid option for lightweight applications.

EfficientNet-B2 and MobileNetV3-Large followed closely with competitive results (F1 ≈ 0.84), maintaining good efficiency.

In contrast, VGG16_bn/VGG19_bn underperformed given their large size, while ShuffleNetV2 and SqueezeNet1.1 lacked sufficient capacity, confirming that higher-complexity models are required for this fine-grained task.

These experiments confirm that high-capacity ImageNet-pretrained models significantly outperform lightweight or domain-mismatched architectures. ConvNeXt-Tiny emerged as the top performer, with ResNet34, ResNeXt50, and EfficientNet-B1 as efficient alternatives depending on resource constraints. This highlights the importance of both model scale and pretraining domain when transferring to fine-grained classification tasks under limited data conditions.

F. Hyperparameter Tuning and Optimization

In the final phase of experimentation, we focused on fine-tuning the top two performing models (ConvNeXt-Tiny and ResNet34) with the goal of maximizing classification performance while maintaining real-time inference capabilities. We conducted an extensive grid-based search over hyperparameters including learning rate, optimizer, weight decay, scheduler type, batch size, dropout probability, and label smoothing. The tuning was carried out systematically to investigate the individual and combined effects of each component.

Both models were trained using cross-entropy loss, combined with early stopping based on validation F1-score. The most promising optimizers were RMSprop, AdamW, and SGD, evaluated with learning rates in the range of 3e-5 to 5e-4. Learning rate schedulers included CosineAnnealingLR, StepLR, and OneCycleLR.

Additional regularization techniques such as dropout (p = 0.0–0.2) and label smoothing (ε = 0.0–0.1) were applied to prevent overfitting. We also tested different layer unfreezing depths to assess the impact of partial versus full fine-tuning.

G. Best Configurations and Results

ConvNeXt-Tiny (FT_C6) achieved the highest overall performance with:

- Test Accuracy: 93.48%
- F1-Score: 0.932
- Inference Time: 0.000396 sec/image
- Configuration: RMSprop, CosineAnnealingLR, 3e-5 learning rate, 0.00005 weight decay, dropout p=0.2, label smoothing ε=0.1, batch size = 32

ResNet34 (FT_R6) performed best among the ResNet34 runs with:

- Test Accuracy: 87.92%
- F1-Score: 0.874
- Inference Time: 0.000735 sec/image
- Configuration: RMSprop, CosineAnnealingLR, 5e-5 learning rate, 0.0001 weight decay, dropout p=0.2, label smoothing ε=0.1, batch size = 16

These results clearly establish ConvNeXt-Tiny as the most effective model, outperforming ResNet34 in every key metric while maintaining fast inference and moderate parameter count.

H. Hyperparameter Sensitivity and Observations

Learning rate was the most sensitive hyperparameter. Very low values (e.g., 1e-5) or mismatched schedulers (e.g., OneCycle with small LR) led to drastic underfitting, as seen in FT_C1 (F1 = 0.20) and FT_R4 (F1 = 0.01).

Regularization proved critical. Dropout (p = 0.2) and label smoothing (ε = 0.1) consistently improved validation and test performance, stabilizing deeper fine-tuning and reducing overfitting.

Layer unfreezing impacted training dynamics. Shallow unfreezing (e.g., FT_C1, FT_C2) yielded near-random performance, while deeper unfreezing with appropriate scheduling (e.g., FT_C6) significantly boosted generalization.

Batch size affected convergence stability: ConvNeXt-Tiny favored 32–48, while ResNet34 performed better with smaller batches (16–24).

The final ConvNeXt-Tiny model (FT_C6) showed:

- Rapid convergence with early stopping in epoch 6.
- A near-diagonal confusion matrix, indicating effective multiclass discrimination.
- Minimal overfitting: training accuracy reached 99%, while validation/test accuracy remained stable.

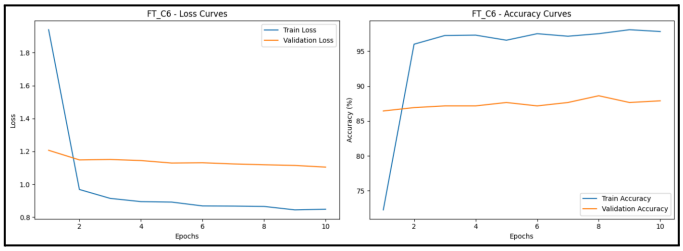
These results highlight the importance of detailed hyperparameter tuning, especially when adapting pretrained models to fine-grained tasks with limited data. The successful application of careful regularization, learning rate control, and progressive unfreezing enabled ConvNeXt-Tiny to achieve state-of-the-art performance within the constraints of real-time inference.

V. RESULTS

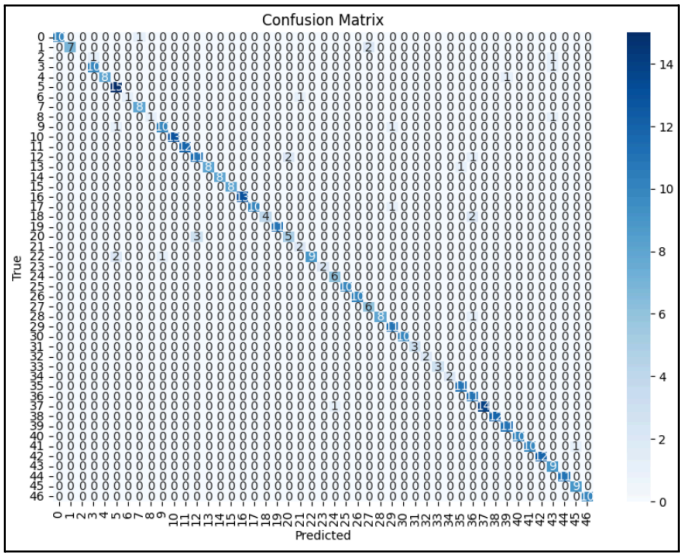
This section presents a structured evaluation of the models developed in this project. We report standard metrics: test accuracy, F1-score, precision, recall, and inference time. Models are compared not only by predictive performance but also by computational cost and deployment suitability. The analysis is supported by visualizations including loss/accuracy curves, confusion matrices, and performance trade-off plots.

Model	**Test Acc.**	**F1 Score**	**Precision**	**Recall**	**s/Image**
ConvNeXt-Tiny (FT_C6)	93.48 %	0.932	0.943	0.935	0.000396
ResNet34 (FT_R6)	87.92 %	0.874	0.886	0.879	0.000735

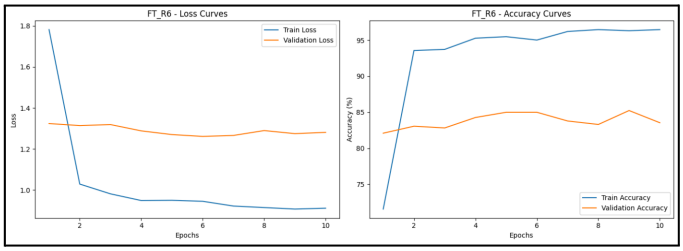
[Figure 1] Summarizes these results, highlighting ConvNeXt-Tiny's superior trade-off between accuracy and inference efficiency.



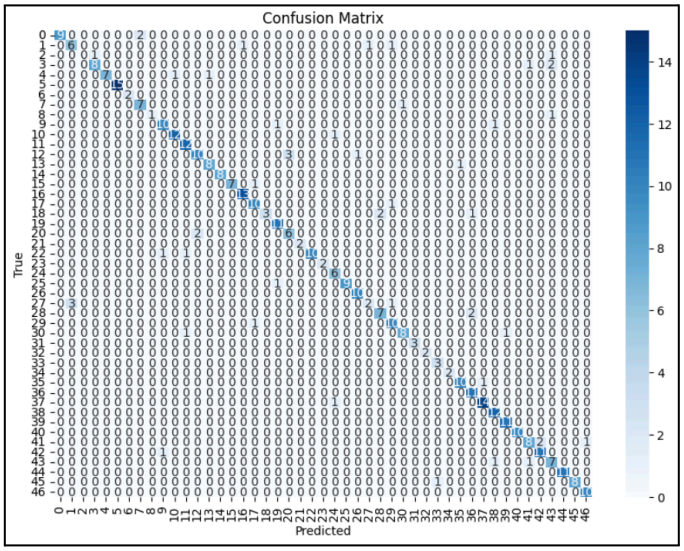
[Figure 2] This figure presents the training and validation loss and accuracy curves for the ConvNeXt-Tiny (FT_C6) model. The model shows rapid convergence within the first few epochs, with stable validation performance and minimal overfitting. These dynamics reflect an effective training strategy and strong generalization under full fine-tuning.



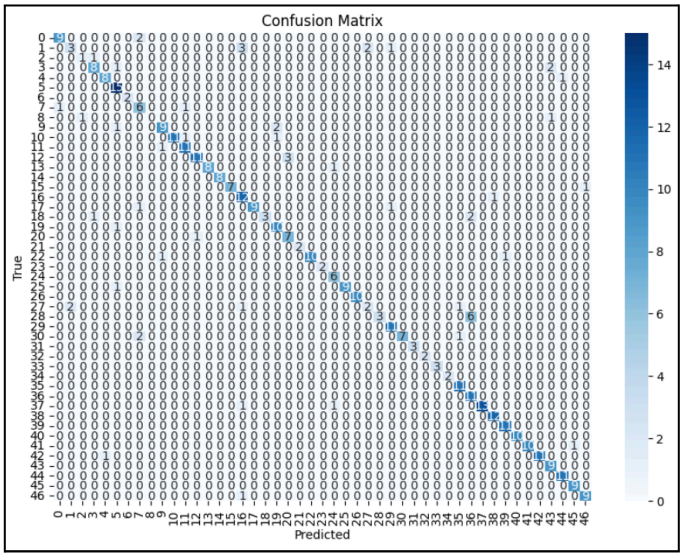
[Figure 3] The confusion matrix for ConvNeXt-Tiny (FT_C6) on the test set reveals a clear diagonal structure, confirming consistent classification accuracy across all 47 classes. The minimal presence of off-diagonal elements indicates that no class was systematically misclassified, highlighting the model's robustness.



[Figure 4] This figure shows the training and validation loss and accuracy curves for ResNet34 (FT_R6). While the model converges smoothly, the validation accuracy plateaus earlier and exhibits higher variance than FT_C6, suggesting reduced generalization and increased susceptibility to minor overfitting.



[Figure 5] The confusion matrix for ResNet34 (FT_R6) demonstrates strong but less consistent classification performance compared to FT_C6. While the diagonal remains prominent, there is an increased concentration of off-diagonal misclassifications, particularly affecting less represented or visually ambiguous classes.



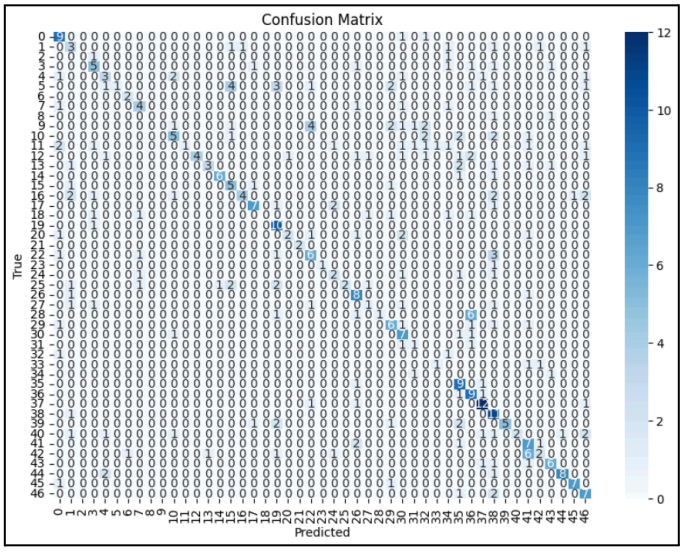
[Figure 6] This matrix illustrates some misclassification patterns for ResNet34 (FT_R6). The diagonal dominance is preserved, yet higher off-diagonal density, relative to FT_C6, points to more frequent confusion between similar poses.

Model	**# Parameters**	**Test Acc.**	**Precision**	**Recall**	**F1-Score**	**Inference Time**
EfficientNet_b1	6,573,391	86.23 %	0.88	0.86	0.86	0.000976 sec/image

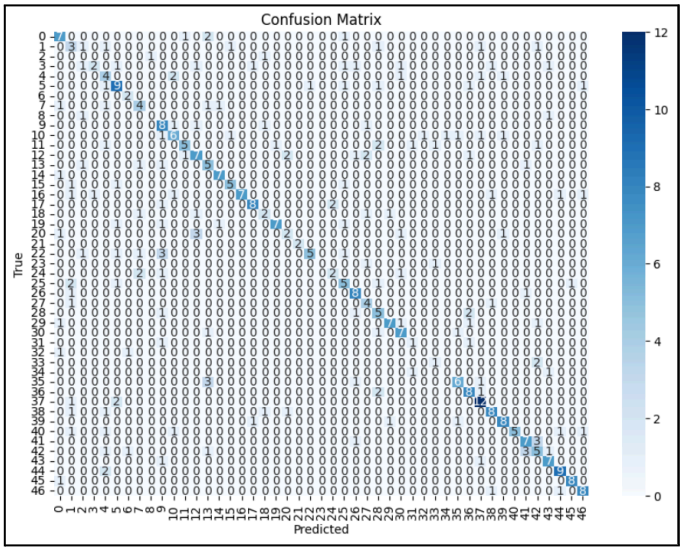
[Figure 7] This figure summarizes the test performance of EfficientNet-B1, which achieved an F1-score of 0.86 and 86.23% test accuracy. Despite slightly slower inference than ConvNeXt-Tiny, the model offers an excellent trade-off between performance and efficiency, especially given its compact architecture (6.57M parameters).

Model	**# Parameters**	**Test Acc**	**Precision**	**Recall**	**F1-Score**	**Inference Time**
ResSimDW_BatchNorm - 1 Layer Unfreeze	208.403	47.48	0.63	0.48	0.48	0.000077 sec/image
ResSimDW_BatchNorm - More than 1 Unfreeze Layer	208,403	57.49	0.61	0.57	0.57	0.000081 sec/image

[Figure 8] The table compares two ResSimDW_BatchNorm variants with different unfreezing strategies. Allowing multiple layers to be trainable significantly improves F1-score and recall, with the multi-layer version reaching 57.49% accuracy. However, both models underperform compared to ImageNet-based models.



[Figure 9] This confusion matrix shows the classification results for the ResSimDW_BatchNorm model with only one layer unfrozen. The weak diagonal structure and widespread off-diagonal elements indicate poor class separation, limited feature adaptation, and overall low generalization capability.



[Figure 10] The matrix displays results for the multi-layer unfreezing variant of ResSimDW_BatchNorm. While diagonal definition improves relative to the single-layer variant, frequent off-diagonal errors persist, particularly between visually similar poses, indicating some adaptability gains, though still insufficient for robust classification.

VI. Discussions & Future Work

A. Discussions

The experimental results establish ConvNeXt-Tiny (FT_C6) as the best-performing architecture, consistently surpassing ResNet34 (FT_R6) across accuracy, F1-score, and inference time. Although both models use pretrained weights and were fine-tuned under similar conditions, ConvNeXt-Tiny's design (combining CNN simplicity with transformer-inspired refinements) likely enables more expressive representations, especially for subtle inter-class differences in fine-grained pose data. Its faster convergence and better regularization (Fig. 1) also reflect training stability and stronger generalization, as confirmed by the clean diagonal of its confusion matrix (Fig. 2).

Regarding hyperparameter tuning, the superior performance of FT_C6 underscores how critical it is to calibrate all training components cohesively. This configuration, combining RMSprop as the optimizer with CosineAnnealing learning rate scheduling, proved particularly effective for stabilizing training and avoiding sharp oscillations during weight updates. The chosen base learning rate (3e-5) was significantly lower than in other tested configurations, allowing for smoother convergence and avoiding early divergence, which was observed in runs like FT_C1 and FT_C2. In those cases, despite having a strong model architecture, inappropriate optimizer choices (e.g., AdamW with large batch size) and mismatched learning rate schedules (e.g., OneCycle with low LR) led to extremely poor generalization, with F1 scores dropping as low as 0.20 or below.

Regularization also played a central role in enabling effective fine-tuning. The use of dropout (p=0.2) helped prevent overfitting during the deeper unfreezing stages, where the model starts adapting early-layer filters to the new domain. Simultaneously, label smoothing (ε=0.1) mitigated overconfidence in predictions, which is especially useful in imbalanced multi-class settings like yoga pose classification, where some classes are represented by fewer than 20 examples. Without label smoothing (as in FT_C3), we observed increased overfitting and a lower F1 score despite relatively strong accuracy, revealing a hidden performance gap.

The choice of batch size = 32 in FT_C6 balanced convergence stability and memory efficiency. Interestingly, batch size influenced performance in a model-specific way—ConvNeXt-Tiny favored medium batch sizes (32–48), while ResNet34 achieved better results with smaller batches (16–24), possibly due to differing optimization dynamics and batch norm behavior.

Moreover, the interaction between components was nontrivial: for instance, using the same RMSprop optimizer with a different scheduler (StepLR instead of CosineAnneal) or altering dropout levels slightly led to measurable performance drops. This indicates that ConvNeXt-Tiny is highly sensitive to the configuration of all training hyperparameters, and that optimal performance is achieved not through isolated tuning of single variables, but through synergistic combinations tuned through iterative experimentation.

In summary, FT_C6's success is not merely architectural, but a direct result of a harmonized tuning strategy—where optimizer, learning rate, scheduler, batch size, and regularization work in concert to maximize generalization. This highlights that in fine-grained classification problems with limited data, hyperparameter tuning is not optional but foundational, and careless choices—even with a state-of-the-art backbone—can lead to model collapse or wasted compute.

Another notable dimension of this work was the deliberate attempt to leverage lightweight CNNs pretrained on SVHN. These models, due to their compactness and prior success in Lab 3, seemed promising for resource-constrained environments. However, the SVHN domain mismatch (digits vs. complex human poses) and image downscaling to 32×32 led to information loss and poor transferability. This was especially evident in frozen extractor experiments, where classification collapsed to a few dominant classes. Yet, when more layers were unfrozen, especially mid-level blocks and the final classifier head, performance significantly improved. For example, ResSimDW_BatchNorm with multiple unfrozen layers reached 57.49% accuracy and F1 = 0.57, compared to only 47.48% and F1 = 0.48 with a single-layer unfreeze. This highlights the importance of deeper adaptation in cross-domain transfer.

The two confusion matrices of ResSimDW_BatchNorm reflect these dynamics. The one-layer unfreeze matrix reveals scattered predictions and weak signal, while the multi-layer version exhibits a clearer diagonal, although still with frequent errors. These models, while inferior in accuracy, maintained exceptional inference speed (~0.00008 sec/image), validating their relevance in real-time settings with tight latency budgets.

A standout runner-up was EfficientNet-B1, which demonstrated balanced performance—F1-score of 0.86 and test accuracy of 86.23%—while maintaining a small parameter count and moderate inference time. Its strong performance despite a smaller capacity further reinforces the value of well-scaled pretrained models from ImageNet over training-from-scratch or domain-mismatched alternatives.

Inference time emerged as a critical metric, particularly in the context of real-time yoga pose feedback systems. Models like ConvNeXt-Tiny and ResSimDW variants achieve sub-millisecond latency, ensuring responsiveness on consumer-grade hardware. This metric often trades off with accuracy and model size, but ConvNeXt-Tiny's results show that it is possible to balance all three with careful architecture and tuning choices.

Finally, an inspection of misclassifications from the best model (FT_C6) suggests semantic confusion between visually similar poses—e.g., slight variations in limb angle, mirrored postures, or transitions. In many cases, the background and angle of the photo also influenced performance. Misclassified images often involved occlusions, unusual camera perspectives, or poses captured mid-transition, all of which introduce inter-class ambiguity that even well-trained models struggle to resolve. This underlines a key limitation of purely image-based approaches and suggests that integrating pose keypoints or temporal context may further boost robustness.

B. Future Work

This project successfully demonstrated the effectiveness of convolutional neural networks (CNNs) for fine-grained yoga pose classification under real-world constraints such as limited data, severe class imbalance, and computational efficiency. Through a rigorous experimental process, we evaluated a broad range of architectures and optimization strategies, leading to the clear identification of ConvNeXt-Tiny as the most performant and efficient model. Notably, ConvNeXt-Tiny achieved 93.48% test accuracy, F1-score of 0.932, and sub-millisecond inference, confirming its readiness for deployment in resource-constrained environments.

A critical insight from our experimentation was the importance of pretraining domain alignment. Lightweight CNNs pre-trained on low-resolution datasets like SVHN failed to generalize effectively, even after fine-tuning, reinforcing that successful transfer learning depends not only on

architecture but also on the relevance of source data. Similarly, naive shallow unfreezing or improper learning rate schedules led to underfitting, while carefully tuned deep fine-tuning with label smoothing and dropout yielded substantial improvements in generalization.

Several directions emerge as promising continuations of this work:

- **Real-Time Webcam Integration**: With ConvNeXt-Tiny's low inference latency, the model is well-suited for real-time applications. A natural next step would be integrating the classifier with a live webcam stream to create an interactive yoga assistant capable of recognizing and correcting user poses in real time. This system could provide immediate feedback, enhancing personal fitness sessions at home.

- **Pose Tracking and Sequence Analysis**: Incorporating temporal models (e.g., RNNs or Transformers) could enable tracking pose transitions and assessing sequence correctness, supporting applications like yoga flow evaluation and exercise routine monitoring.

- **Expanding to Full-Body Exercise Monitoring**: Beyond yoga, the methodology could generalize to other fitness domains such as pilates, strength training, or physiotherapy routines. Adapting the classifier to recognize full-body exercise postures would support broader use cases in health and sports.

- **Data Augmentation via Synthetic Generation**: Given the dataset's size and imbalance limitations, one exciting research avenue is the use of generative models (e.g., GANs or diffusion models) to create synthetic samples for rare poses. This could substantially increase data diversity and improve classifier robustness without costly manual annotation.

In summary, the results and insights obtained lay a strong foundation for advancing yoga pose recognition into interactive, real-time, and mobile applications. With further refinements and system integration, this work can serve as a core component of intelligent fitness platforms that offer personalized, immediate, and scalable coaching experiences.

VII. Manual Prediction

To further validate the performance of our best model in real-world scenarios, we conducted a manual prediction using an external image of one of the authors performing a yoga pose. This image was not part of the training, validation, or test splits, making it a reliable test of the model's generalization to unseen, real-world inputs.

The image was processed using the same normalization and resizing pipeline applied during model evaluation to ensure consistency. We loaded the final fine-tuned ConvNeXt-Tiny (FT_C6) checkpoint and performed inference using PyTorch in evaluation mode. The model successfully predicted the correct yoga pose class from the image with high confidence.

This test confirms that the model is not only effective on curated dataset samples but also capable of robust classification when faced with real photographs under non-standard conditions (e.g., different lighting, clothing, or background). The result supports its potential for integration into practical applications such as personalized training assistants or pose-correction feedback systems.

[Figure 11] Real-world photo used for manual prediction. The image features one of the authors performing a yoga pose in a home environment, with natural lighting, background objects, and casual clothing. Despite these non-ideal conditions, the model correctly predicted the pose, demonstrating strong generalization and robustness to real-world variability.

VIII. Conclusions

This project explored the challenging task of multiclass yoga pose recognition using convolutional neural networks under realistic constraints (limited data, class imbalance, and a need for real-time inference). Through systematic experimentation, we evaluated baseline CNNs, lightweight architectures, and state-of-the-art pretrained models to assess their performance, efficiency, and generalization on a fine-grained classification task.

Our most significant finding was the superior performance of ConvNeXt-Tiny, which achieved a 93.48% test accuracy and 0.932 F1-score, while maintaining a sub-millisecond inference time. Its success stemmed not only from its architecture but from careful tuning, emphasizing the critical role of harmonized optimization, including scheduler selection, dropout, label smoothing, and unfreezing depth.

Another key insight was the importance of pretraining domain alignment. Lightweight CNNs pretrained on unrelated tasks (like SVHN digits) failed to generalize effectively to human poses (even with partial fine-tuning) highlighting the limitations of naive transfer learning and the need for domain-relevant pretraining in fine-grained applications.

We also observed that training stability and GPU efficiency were highly dependent on architectural choices. While ResNet34 achieved solid results, ConvNeXt-Tiny converged faster and used resources more effectively, showing that modern CNNs can offer both expressivity and deployability without sacrificing performance.

Finally, by combining effective data augmentation, class rebalancing, and advanced fine-tuning strategies, we demonstrated that accurate yoga pose recognition is feasible even under data-scarce conditions. These results lay the groundwork for extending this system into real-time applications, supporting intelligent fitness assistants and adaptive coaching tools.

This project not only advanced our understanding of CNN-based pose recognition but also provided valuable practical experience in deep learning experimentation, hyperparameter tuning, and trade-off analysis between accuracy, speed, and generalization. The lessons learned here have broader implications for building robust, scalable vision systems in real-world, resource-constrained environments.

IX. References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.

[2] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510–4520.

[3] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 6105–6114.

[4] Ma, Y., Shen, X., & Wang, Y. (2022). Fine-grained yoga pose classification using Vision Transformers. *Pattern Recognition Letters*, 155, 37–44. https://doi.org/10.1016/j.patrec.2022.01.002

[5] tr1gg3rtrash. (2021). *Yoga posture dataset* [Kaggle dataset]. https://www.kaggle.com/datasets/tr1gg3rtrash/yoga-posture-dataset

[6] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11976–11986.

[7] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 8026–