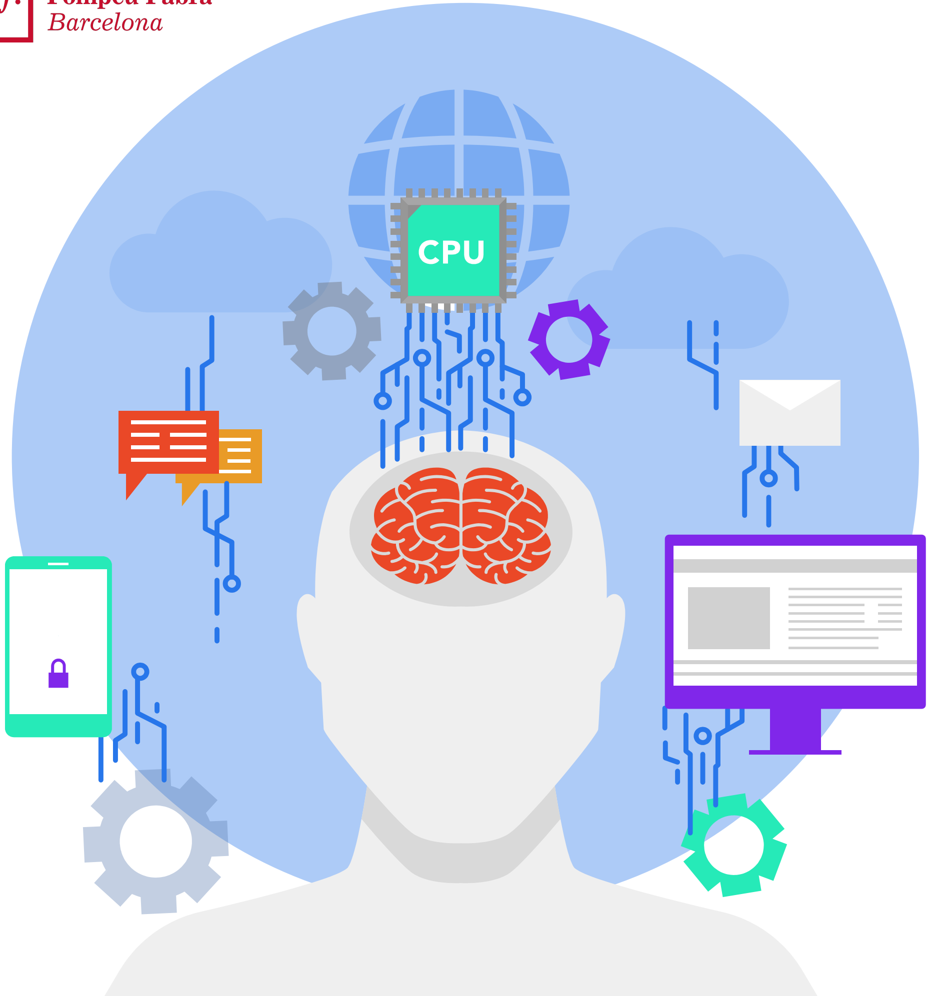
upf. Universitat Pompeu Fabra Barcelona



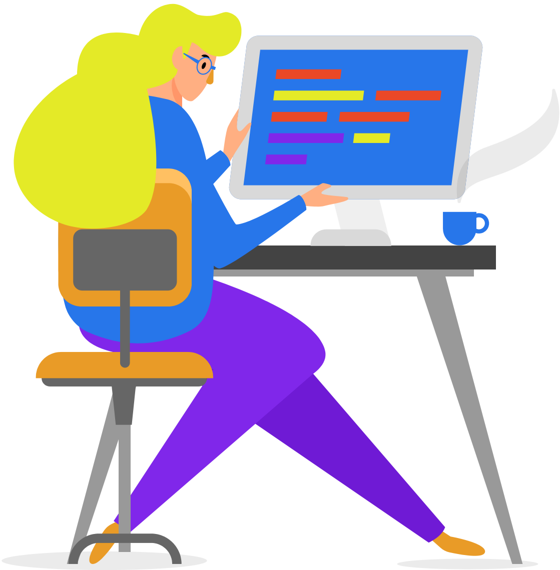
Yoga Pose Recognition Using CNNs

Joan Company & Adrià Cortés

Deep Learning

02/06/2025

Project Overview



01 **Topic Motivation**
Real-world application in fitness & pose correction

02 **Dataset**
Selection & Preparation

03 **Network Architecture**
Model Selection & Description

04 **Hyperparameter Tuning**
Optimizing Accuracy, Convergence and # Parameters

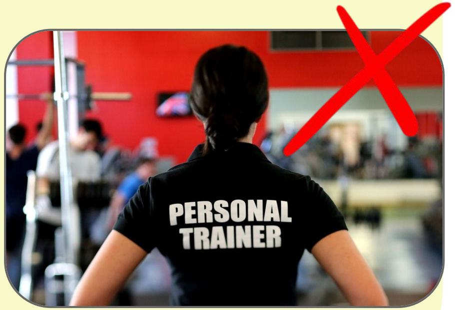
05 **Model Evaluation**
Visualizations & Predictions

06 **Conclusions & Future Implications**

Project Motivation

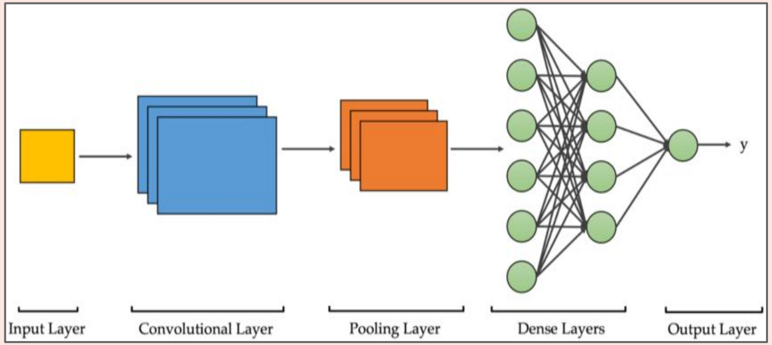
Health Motivation

- Promoting Healthy Lifestyle through AI.
- Democratizing Wellness with Computer Vision.



Technical Motivation

- Apply Deep Learning Concepts from Theory and Labs.
- Transfer Learning with Pre-Trained CNNs

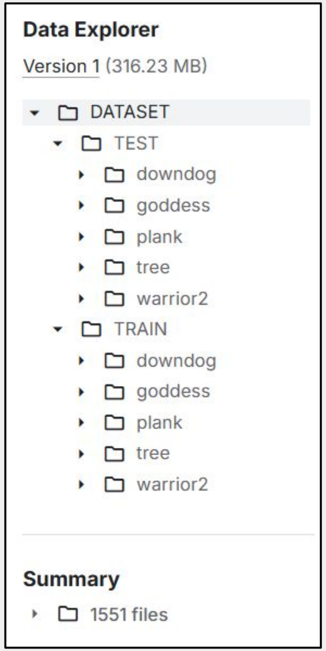


Dataset Selection

kaggle

Option 1: *Yoga Poses Dataset*

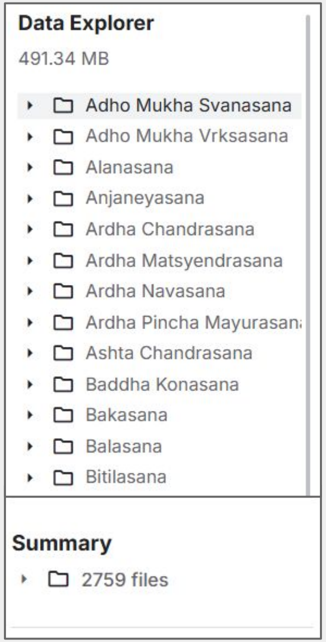
- Well Organized & Clear Split.
- Only 5 Different Classes.
- Different File Formats.
- Watermarks & Text.
- 1551 Sample Images.



[1]

Option 2: *Yoga Posture Dataset*

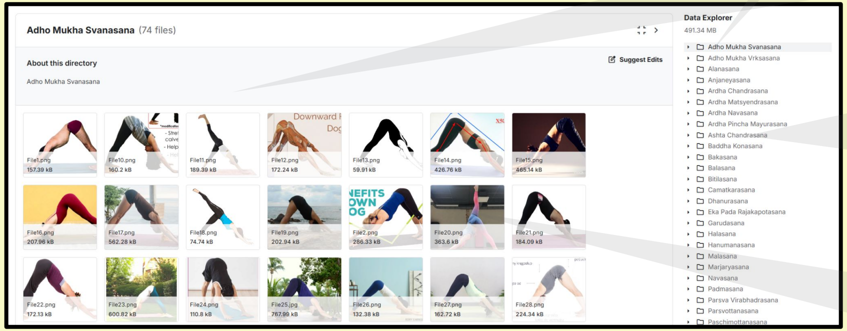
- Well Organized Split.
- 47 Different Classes.
- Compatible File Formats: .png
- 2759 Sample Images



[2]

Dataset Selection: Final Choice

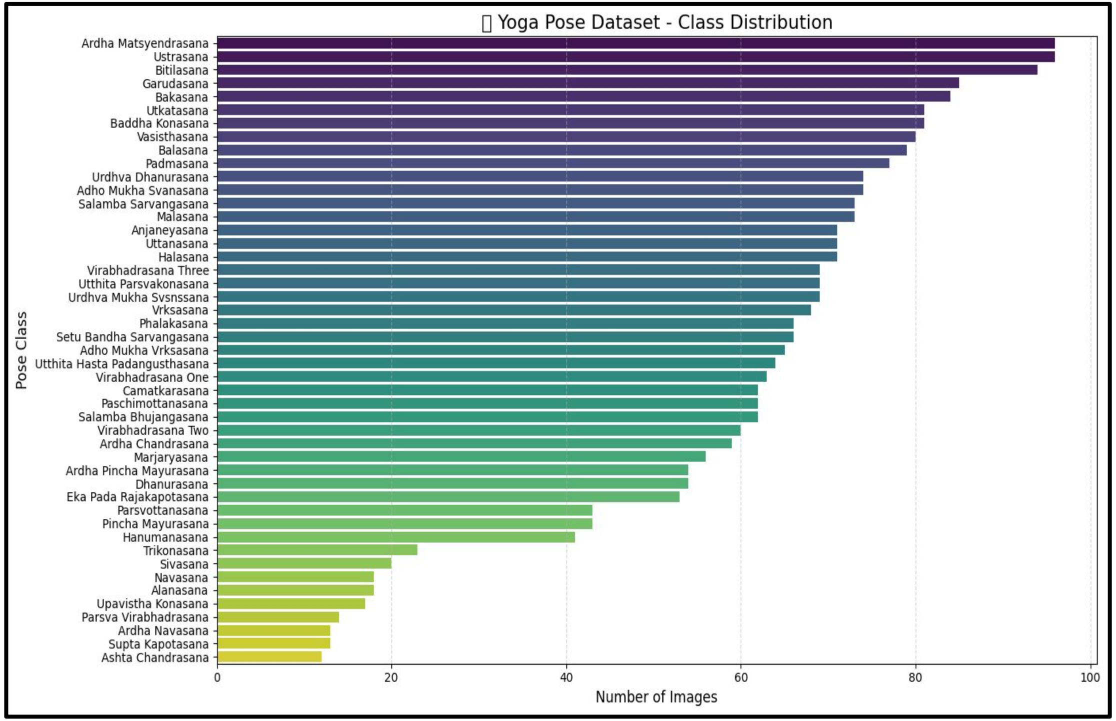
kaggle



- Real-world Yoga Poses Across 47 Categories
- ~ 3,000 Labeled Images With Class Folders
- Includes English & Sanskrit Pose Names
- Suitable for Deep Learning Classification → CNNs

Dataset Preparation

We found **significant imbalance** across 47 yoga poses. To fix this, we applied **class weights in the loss function.**



- Focus More on Underrepresented Classes.
- Counter For Each Yoga Pose Class.
- **Weights** → (1 / Class Frequency)

$$w_i \propto \frac{1}{f_i}$$

- **Random Weighted Sampling** → + Balancing!

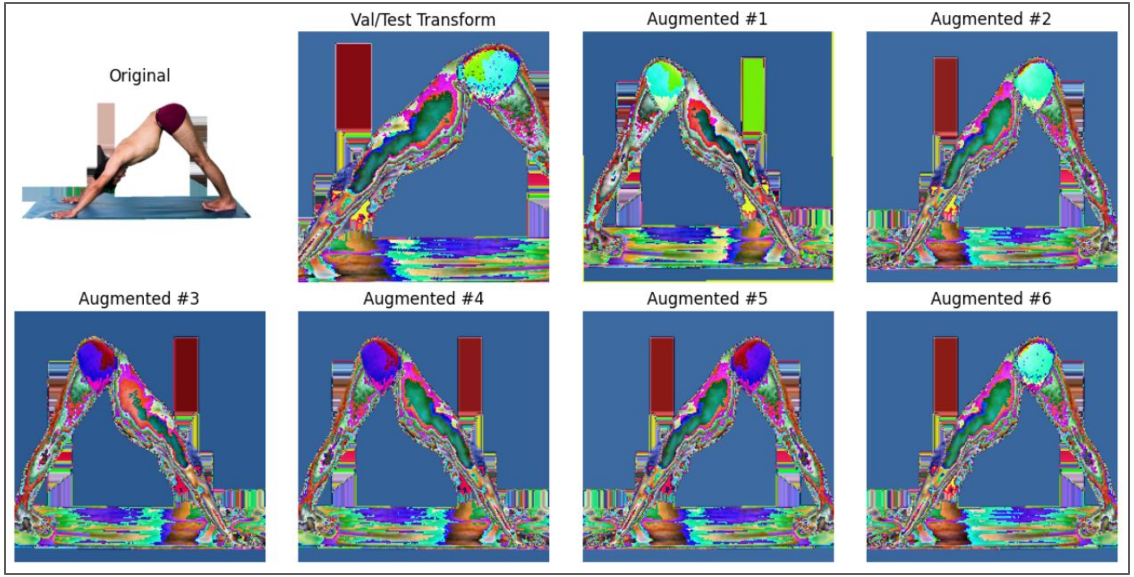
Dataset Preparation

- Applied Data Augmentation on Train Images:

  - Random Crops
  - Flips
  - Perspective Changes

- Stratified Split → **70% / 15% / 15%**

- Consistent Resizing and Normalization to ensure fair evaluation.



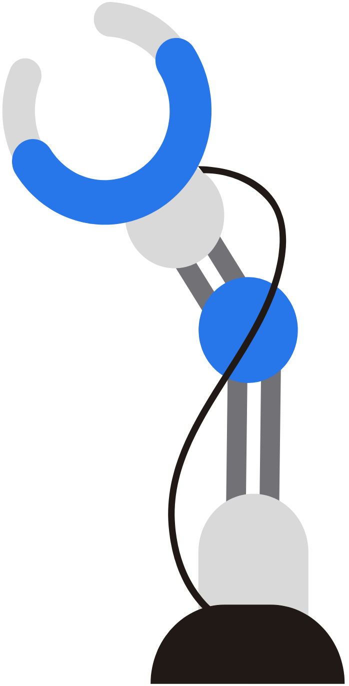

```
train_dataset = ImageDatasetWithTransform(train_samples, train_transforms)
val_dataset   = ImageDatasetWithTransform(val_samples, val_test_transforms)
test_dataset  = ImageDatasetWithTransform(test_samples, val_test_transforms)
```

Network Architecture

1. Simple CNN
2. **Experimentation with CNN: Several CNN architectures pretrained with ImagNet**
3. Modifications to CNN
4. Lightweight CNN
5. **Final Model**
6. **Hyperparameter Tuning**

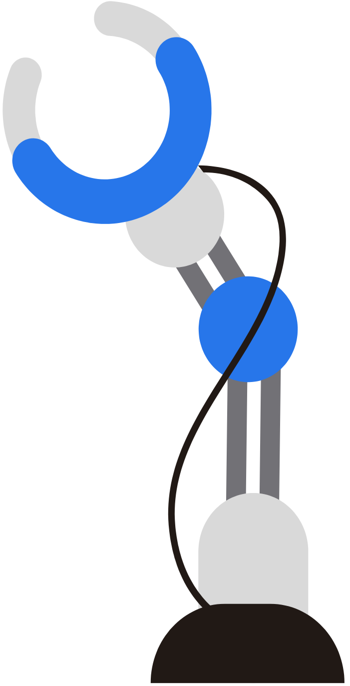
Experimentation on CNN

Model	Parameters	Train Acc	Val Acc	Test Acc	Precision	Recall	F1-Score	Inference Time (sec/image)
resnet18	11,200,623	98.70%	75.06%	77.29%	0.81	0.77	0.77	0.000240 sec/image
resnet34	21,308,783	98.39%	78.69%	80.19%	0.83	0.8	0.8	0.000381 sec/image
resnet50	23,604,335	98.76%	74.58%	76.57%	0.8	0.77	0.76	0.000653 sec/image
wide_resnet50_2	66,930,543	98.60%	71.91%	79.95%	0.84	0.8	0.79	0.000486 sec/image
resnext50_32x4d	23,076,207	98.70%	78.21%	81.16%	0.83	0.81	0.81	0.000642 sec/image
densenet121	7,002,031	98.03%	80.63%	78.99%	0.81	0.79	0.78	0.002226 sec/image
densenet169	12,562,735	98.70%	76.76%	81.88%	0.85	0.82	0.81	0.001570 sec/image
efficientnet_b0	4,067,755	98.70%	77.24%	79.95%	0.82	0.8	0.79	0.000807 sec/image
efficientnet_b1	6,573,391	97.87%	78.21%	81.40%	0.84	0.81	0.81	0.001108 sec/image
efficientnet_b2	7,767,217	98.29%	79.42%	81.88%	0.85	0.82	0.82	0.001748 sec/image
vgg16_bn	134,461,551	96.47%	73.12%	72.71%	0.79	0.73	0.73	0.000199 sec/image
vgg19_bn	139,773,807	97.36%	71.19%	73.67%	0.78	0.74	0.73	0.000224 sec/image
mobilenet_v2	2,284,079	97.56%	74.33%	77.29%	0.8	0.77	0.77	0.000526 sec/image
mobilenet_v3_large	4,262,239	98.65%	72.40%	78.50%	0.81	0.79	0.78	0.000541 sec/image
shufflenet_v2_x1_0	1,301,779	60.60%	35.35%	41.06%	0.63	0.41	0.37	0.000644 sec/image
squeezenet1_1	746,607	87.45%	55.45%	54.59%	0.59	0.55	0.53	0.000256 sec/image
convnext_tiny	27,856,271	98.91%	85.23%	86.71%	0.89	0.87	0.86	0.000438 sec/image

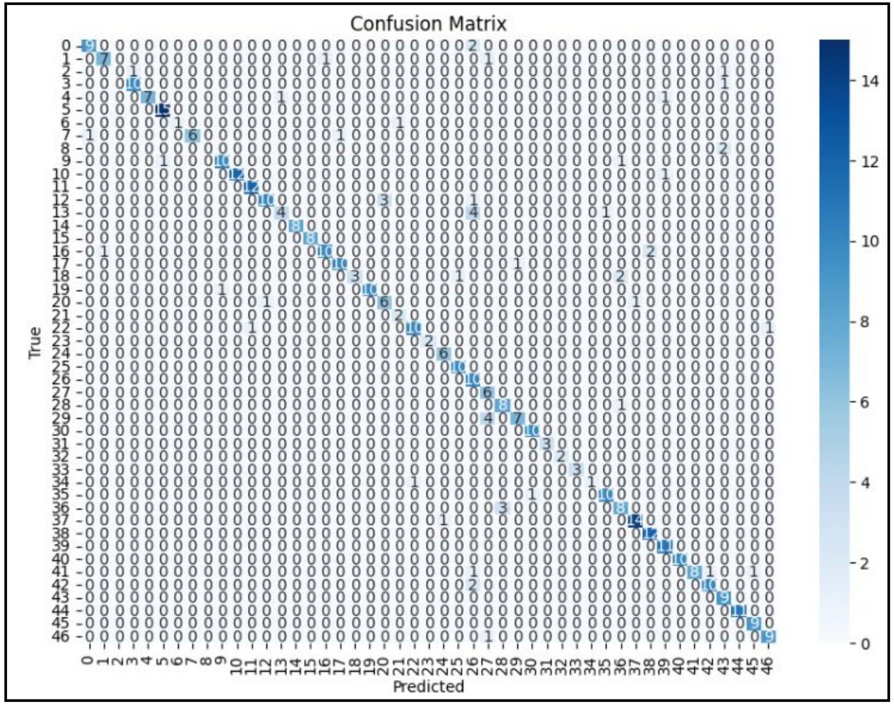


Experimentation on CNN

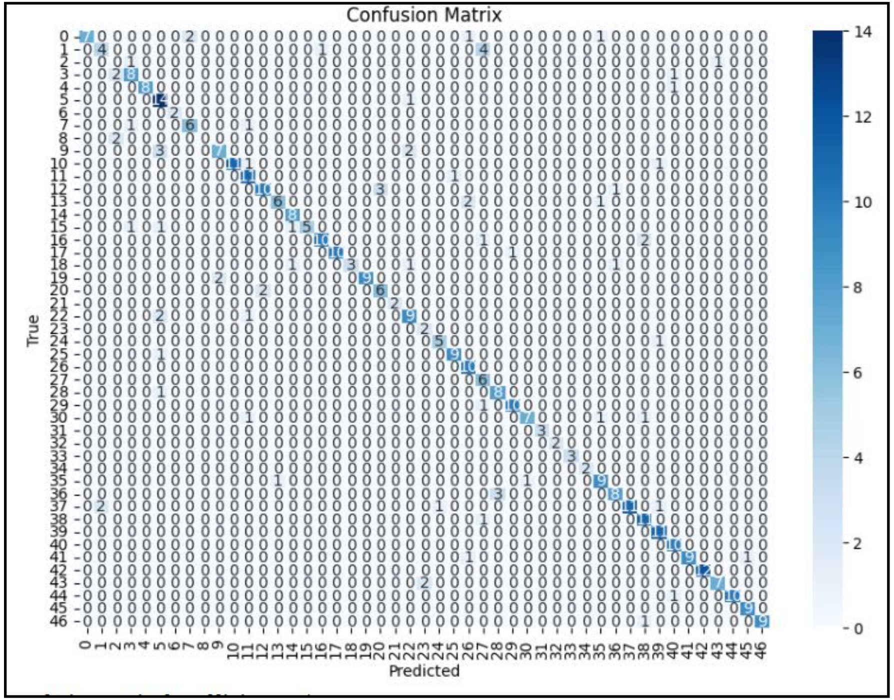
Best Models	Test Accuracy	F1 score	Time s/image	# param.
ConvNeXt Tiny	86.71%	0.86	0.000438	27.9 million
efficientnet_b2	81.88%	0.81	0.001748	7.7 million
densenet121	78.99%	0.78	0.002226	7.002.031
densenet169	81.88%	0.81	0.001570	12.5 million
efficientnet_b0	79.95%	0.79	0.000807	4 million
efficientnet_b1	81.40%	0.81	0.001784	6.5 million
resnext50_32x4d	81.16%	0.81	0.000642	23 million



Experimentation on CNN: Confusion Matrices

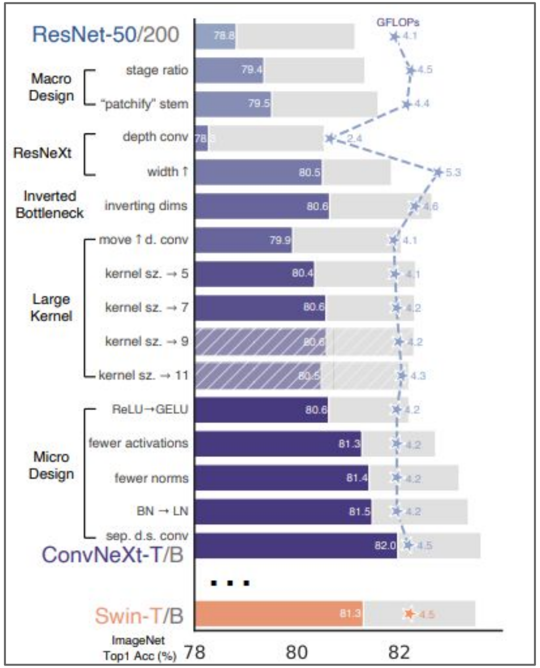


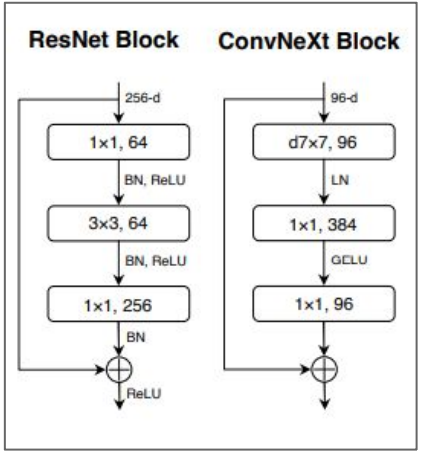
ConvNeXt Tiny

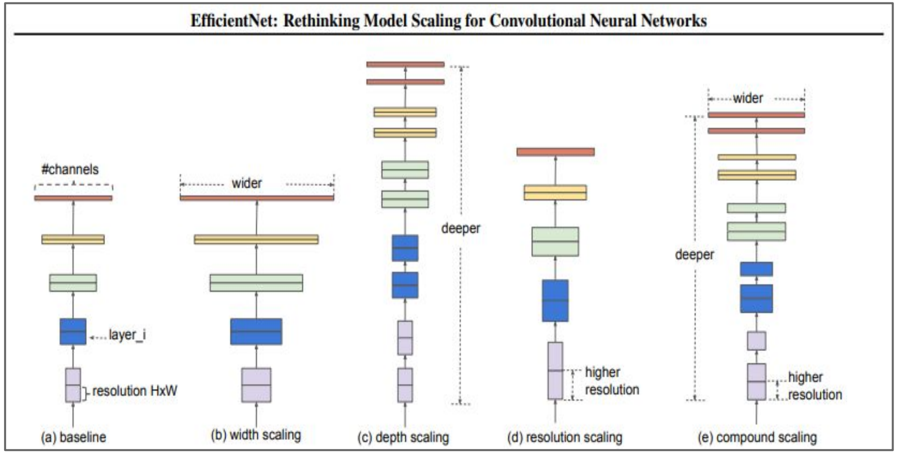


efficientnet_b2

Network Architecture Choice



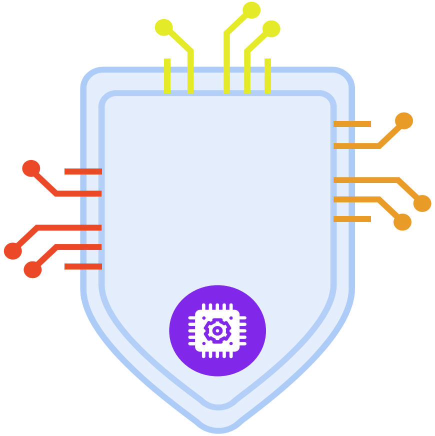




Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Hyperparameter Tuning

```python
# -------- 1) FINETUNE GRID (ONLY CLASSIFIER UNFROZEN) --------
finetune_grid = [
    # (run_name, model_name, optimizer, base_lr, weight_decay, scheduler, batch_size, dropout_p, label_smoothing)
    ("FT_C1", "ConvNeXt-Tiny", "AdamW",  1e-5, 1e-4, "CosineAnneal", 64, 0.2, 0.1),
    ("FT_C2", "ConvNeXt-Tiny", "AdamW",  5e-5, 5e-5, "OneCycle",     64, 0.2, 0.1),
    ("FT_C3", "ConvNeXt-Tiny", "AdamW",  1e-4, 5e-5, "StepLR",       48, 0.2, 0.0),
    ("FT_C4", "ConvNeXt-Tiny", "SGD",     5e-4, 1e-4, "CosineAnneal", 48, 0.2, 0.1),
    ("FT_C5", "ConvNeXt-Tiny", "SGD",     1e-4, 5e-5, "OneCycle",     32, 0.0, 0.1),
    ("FT_C6", "ConvNeXt-Tiny", "RMSprop", 3e-5, 5e-5, "CosineAnneal", 32, 0.2, 0.1),
    ("FT_E1", "EfficientNet-B2", "SGD",   5e-5, 5e-5, "CosineAnneal", 32, 0.2, 0.1),
    ("FT_E2", "EfficientNet-B2", "SGD",   3e-5, 1e-4, "OneCycle",     24, 0.2, 0.1),
    ("FT_E3", "EfficientNet-B2", "SGD",   1e-4, 1e-4, "StepLR",       16, 0.2, 0.0),
    ("FT_E4", "EfficientNet-B2", "AdamW", 3e-5, 5e-5, "OneCycle",     24, 0.2, 0.1),
    ("FT_E5", "EfficientNet-B2", "AdamW", 1e-4, 1e-4, "CosineAnneal", 24, 0.0, 0.1),
    ("FT_E6", "EfficientNet-B2", "RMSprop",5e-5, 1e-4, "CosineAnneal",16, 0.2, 0.1),
]
```



- **Optimizer: Adam, SGD, ...**
- **Learning Rate: [5e-5 ; 1e-4]**
- **Weight Decay: [5e-5 ; 1e-4]**

- **Scheduler**
- **Batch Size: [16 ; 64]**
- **Dropout: [0.0 , 0.2]**

- **Label Smoothing: [0.0 , 0.1]**

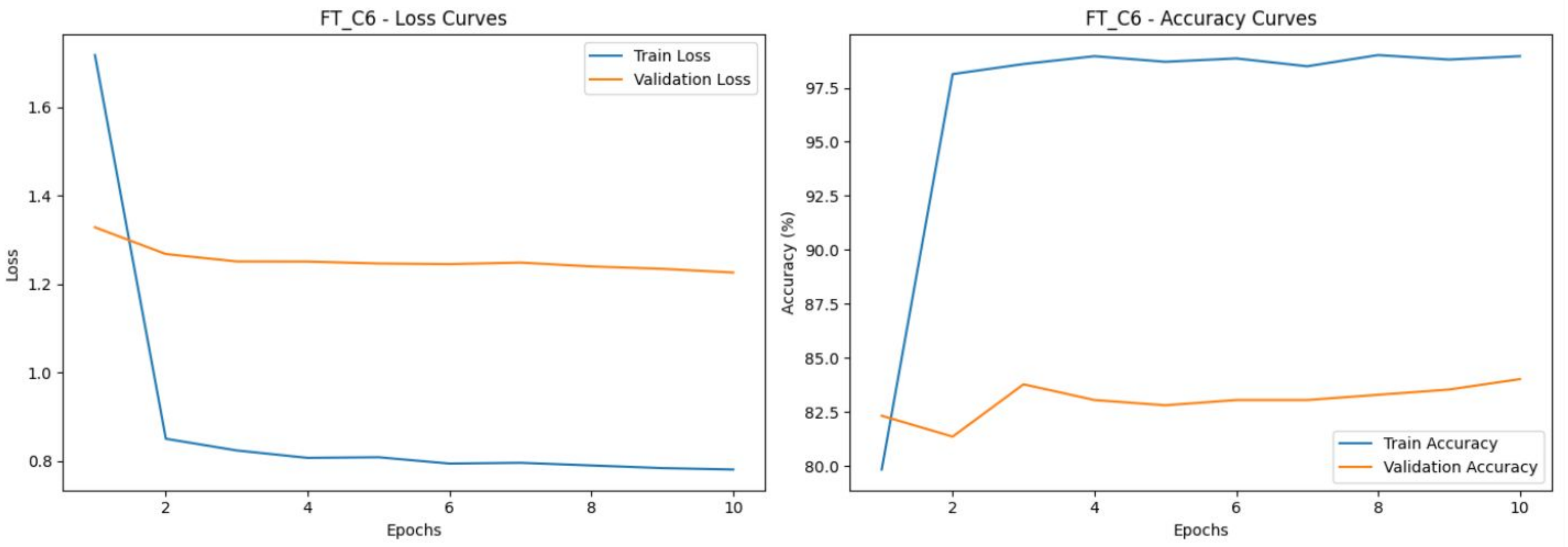
Hyperparameter Tuning

Model Configurations	Test Accuracy	F1 score	Time s/image	# param.
FT_C3	85.27%	0.849	0.2625s	27.9 million
FT_C5	86.71%	0.863	0.0004s	27.9 million
FT_C6	87.20%	0.867	0.000449s	27.9 million

“We trained the model with 6 different configurations and this were the ones showing better performance results.”

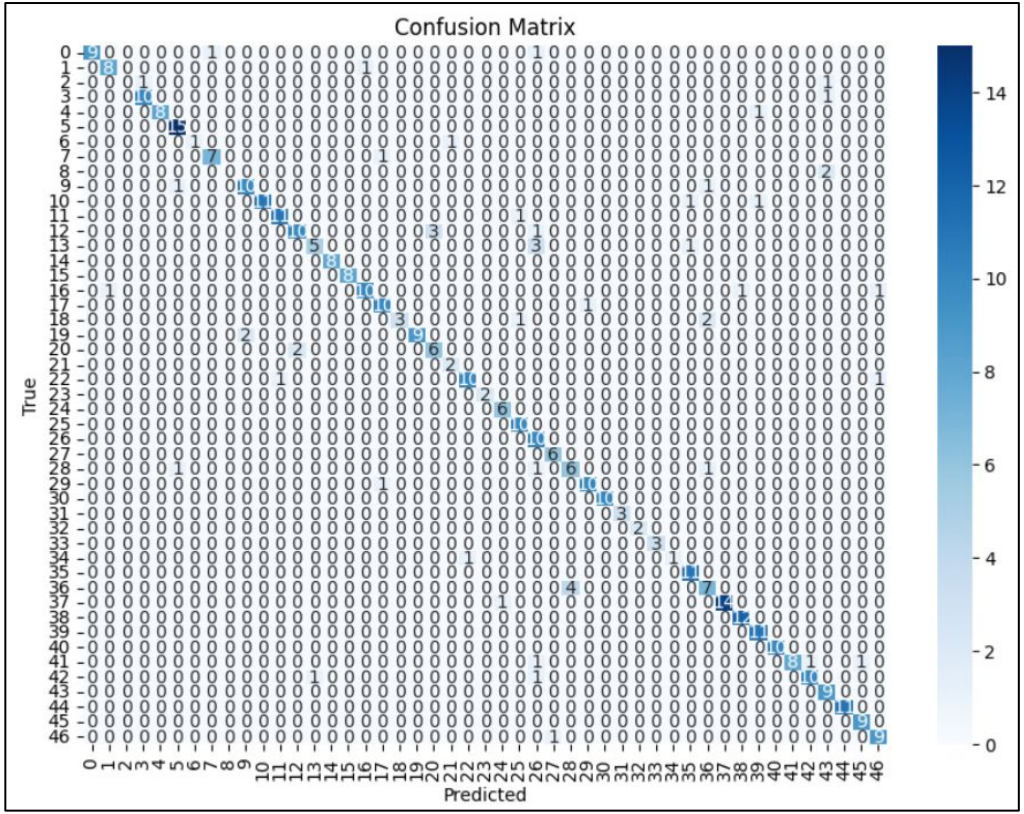
```
# (run_name, model_name, optimizer, base_lr, weight_decay, scheduler, batch_size, dropout_p, label_smoothing)
("FT_C3", "ConvNeXt-Tiny", "AdamW",   1e-4, 5e-5, "StepLR",        48, 0.2, 0.0)
("FT_C5", "ConvNeXt-Tiny", "SGD",     1e-4, 5e-5, "OneCycle",      32, 0.0, 0.1)
("FT_C6", "ConvNeXt-Tiny", "RMSprop", 3e-5, 5e-5, "CosineAnneal", 32, 0.2, 0.1)
```

Model Evaluation & Visualizations



FT_C6 | ConvNeXt-Tiny

Model Evaluation & Visualizations



FT_C6 | ConvNeXt-Tiny

References

[1] Niharika41298. (n.d.). *Yoga poses dataset*. Kaggle. https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset

[2] Tr1gg3rtrash. (n.d.). *Yoga posture dataset*. Kaggle. https://www.kaggle.com/datasets/tr1gg3rtrash/yoga-posture-dataset

[3] Dosovitskiy, A., & Springenberg, J. T. (2022). *Exploring the limits of transfer learning with a unified image classification benchmark*. arXiv. https://arxiv.org/abs/2201.03545?utm_source

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2019). *Identity mappings in deep residual networks*. arXiv. https://arxiv.org/abs/1905.11946?utm_source

[5] PyTorch Contributors. (n.d.). *ConvNeXt Tiny*. PyTorch. https://docs.pytorch.org/vision/main/models/generated/torchvision.models.convnext_tiny.html?utm_source

Thanks For Your Attention!

Joan Company & Adrià Cortés

Deep Learning

02/06/2025