Yoga Poses Classification

Aniket Chandak
Ashraf Saber
David Bui
Deanne Kshipra Charan
Hardik Kumar
Siddartha Thentu

Pose Estimation

- Pose estimation is identifying human poses by locating human joints in the data
- Pose estimation is a vast problem and consist of many sub-problems
  - 2D and 3D pose estimation
  - Single person or Multi-Person pose estimation
  - Dealing with video and image data

Recap on literature review on Pose Estimation

2D Pose Estimation		**3D Pose Estimation**	
2D Single Person	**2D Multi Person**	**3D Single Person**	**3D Multi Person**
● Cascade Feature Aggregation (CFA) ● Video: motion transfer between a highly skilled dancer and an ordinary target subject ● HRNet	● Crowd Pose ● HRNet	● SMPLify ● 3D Single Person with Video	DetectNet + RootNet + PoseNet

Our Project: YogAI model

- Problem Statement: Artificial Intelligence to identify human pose and based on that guess the identical yoga pose.

- Recommend user with a correct posture to help with Yoga

Yoga Pose estimation

- Yoga poses classification

Warrior 1

Tree

Warrior 2

DataSet

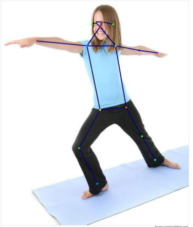
- Consist of Yoga Pose images
- Labelled data
- 8 Classes (Yoga positions) considered for our training
- 33% testing 67% for training

Architectual overview:

Phase 1 (Feature extraction and Training model)

IMAGES → HRNet Inference → Get key points → Normalise & Vectorise → Save Label and data → Train MLP model

Yoga pose

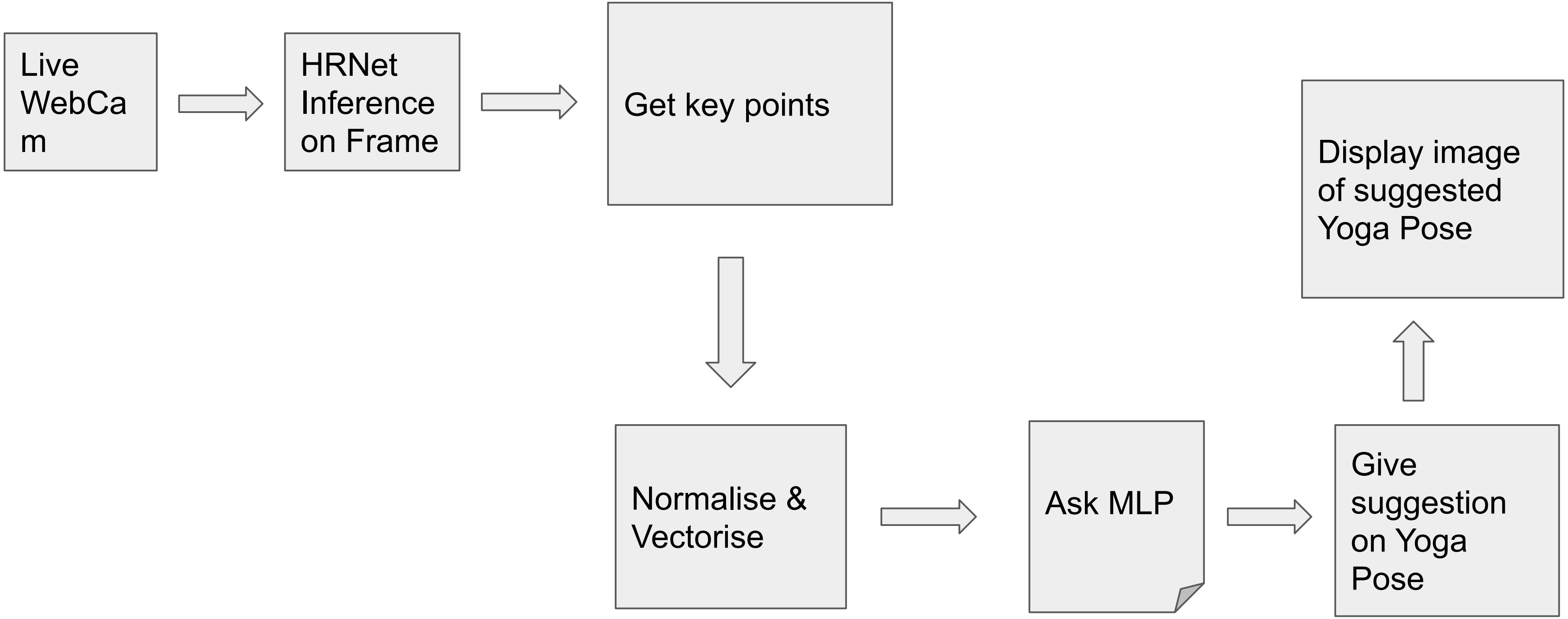


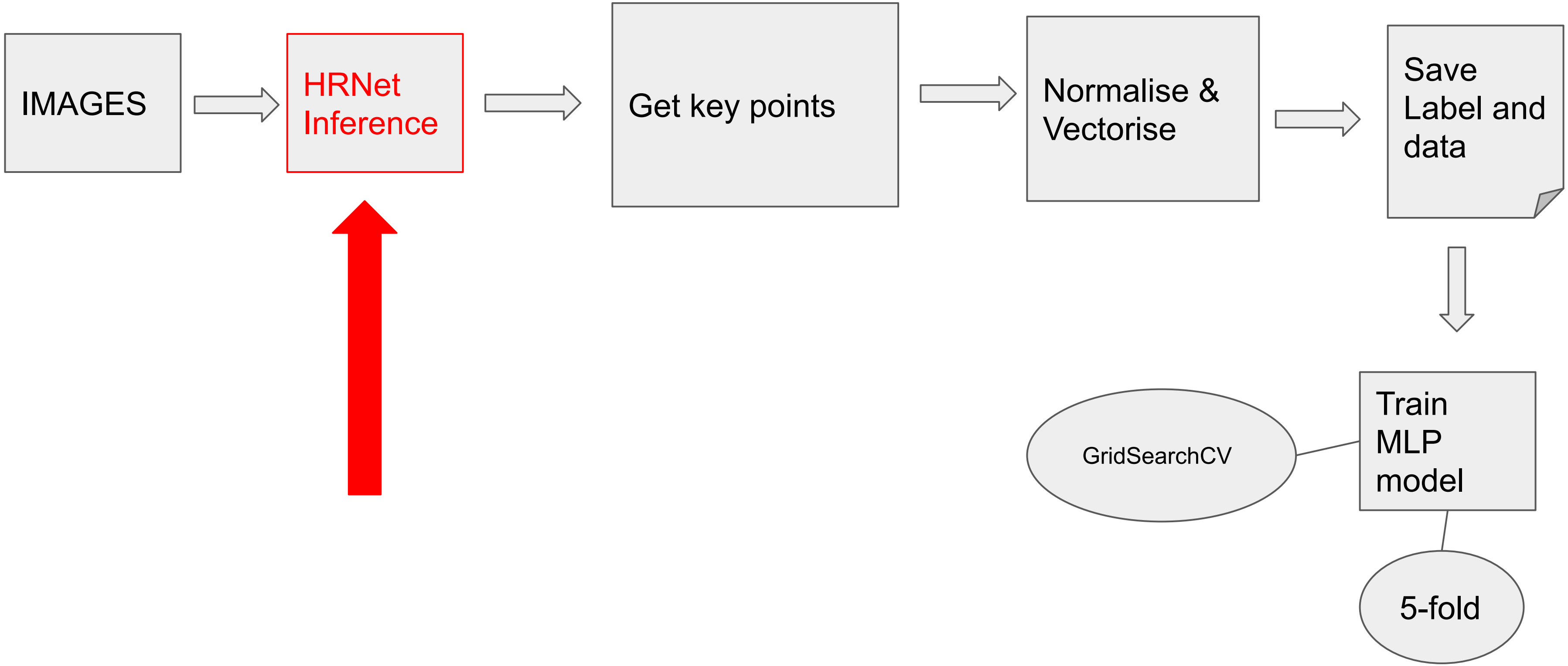
GridSearchCV

5-fold

Architectural overview:

Phase 2 (Application)




IMAGES
HRNet Inference
Get key points
Normalise & Vectorise
Save Label and data
GridSearchCV
Train MLP model
5-fold

Recap Milestone 2.5: SOTA POSE ESTIMATORS

- **HRNET:** High Resolution Network
- **CPN:** Cascade Pyramid Network
- **SHG:** Stacked Hourglass - No code provided
- **Open Pose**

Which SOTA to consider?

- High Accuracy
- Faster

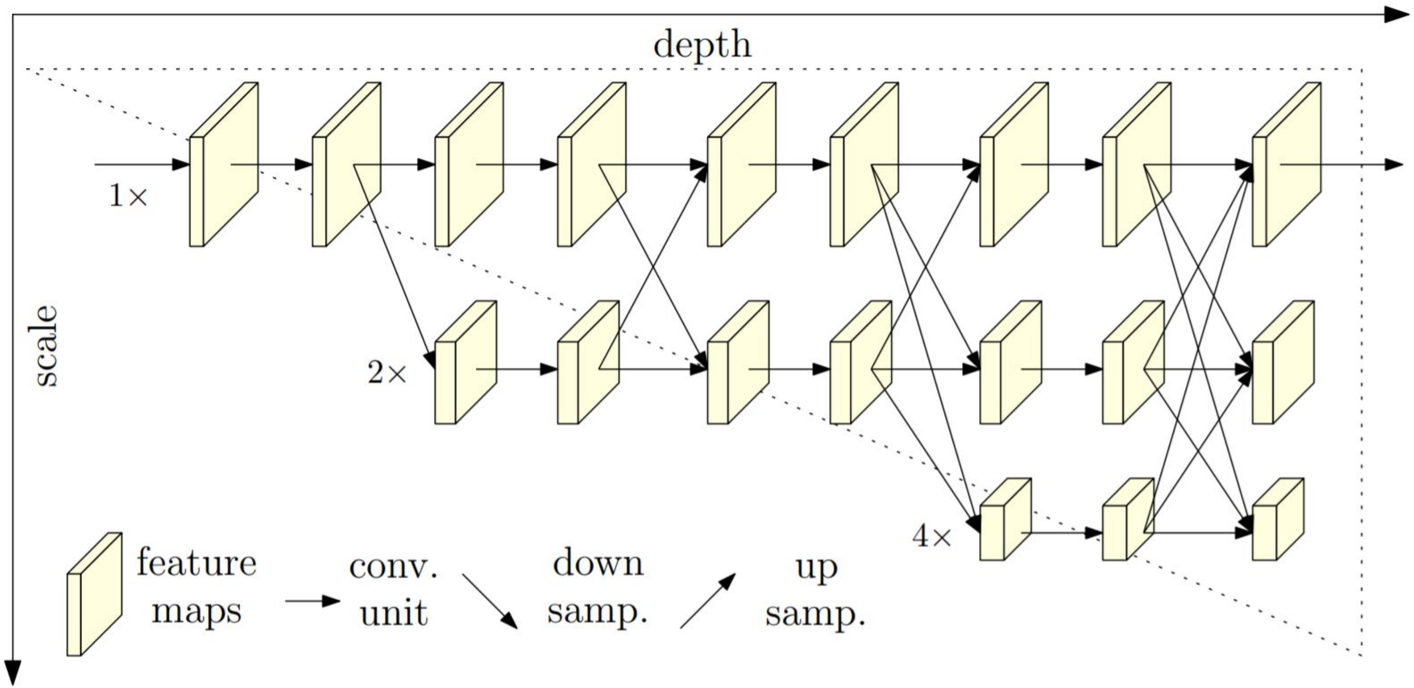
We surveyed with all the models to find out.

Recap Milestone 2.5:

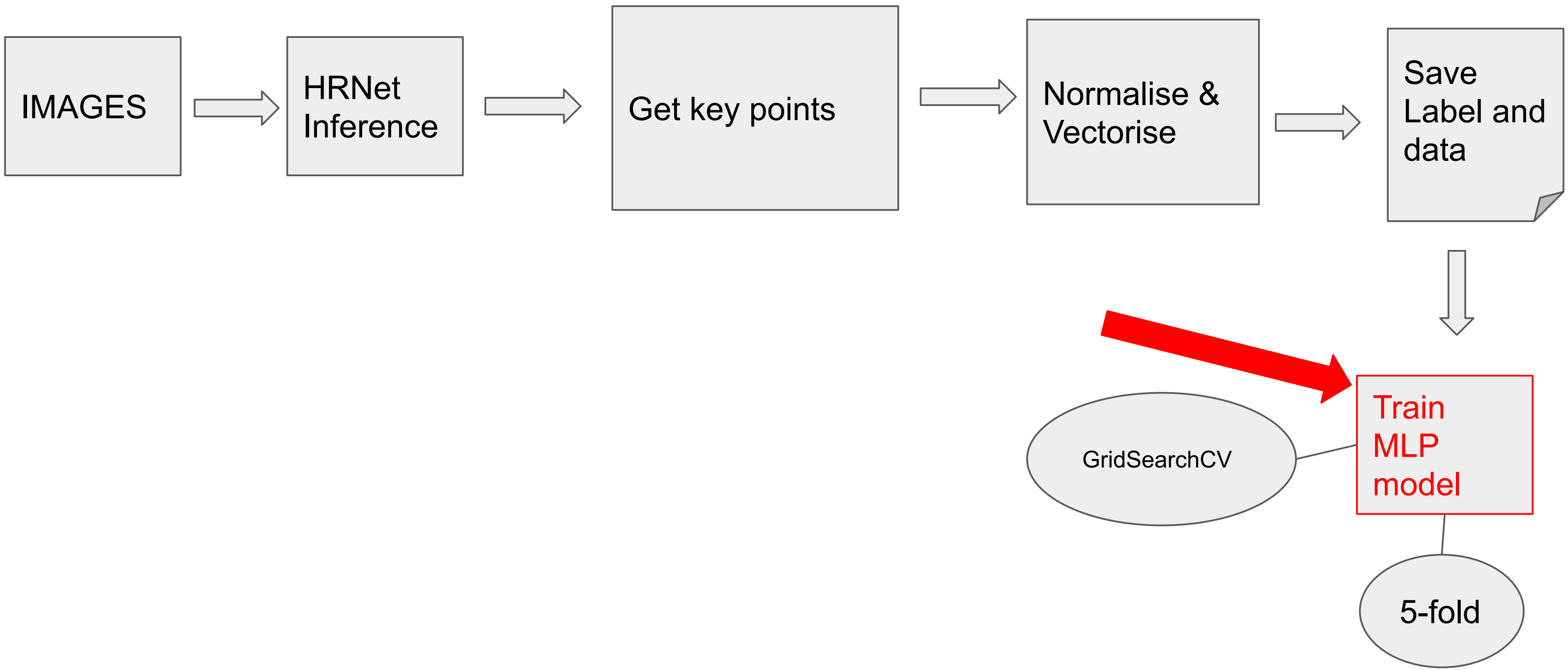
	HRNeT	CPN	OpenPose
Precision :	74.4	71.2	69.7
Inference :	Yes	No	Yes

HRNet vs Open Pose: both provided inference and good accuracies.

Our Pose Estimator: HR-Net



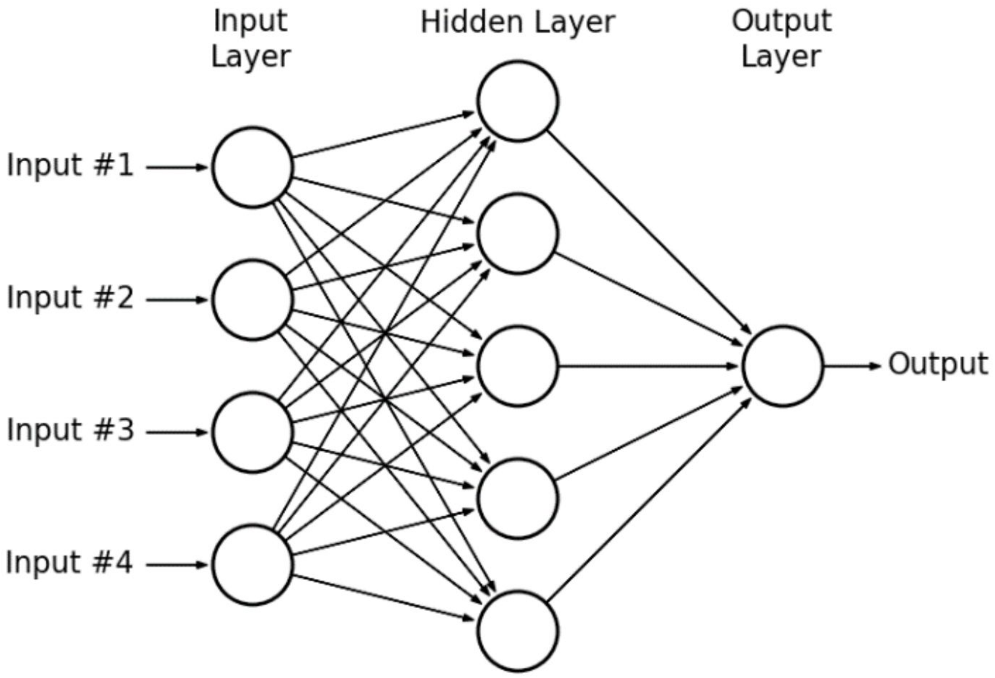
HR Net Architecture


IMAGES
HRNet Inference
Get key points
Normalise & Vectorise
Save Label and data
Train MLP model
GridSearchCV
5-fold

Quick view on MLP

- What is MLP?
- Multi-Layer Perceptron
  - Input Layer (1)
  - Hidden Layer (n) (3 L, 30 Neurons)
  - Output Layer (1)
- Classification
- Feedforward
- We want to train an MLP

  With high accuracy and reliability



MLP

https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

But how to train best and reliable MLP?

- We have used GridSearchCV and 5-fold validation in our code
- Lets see why and how in details

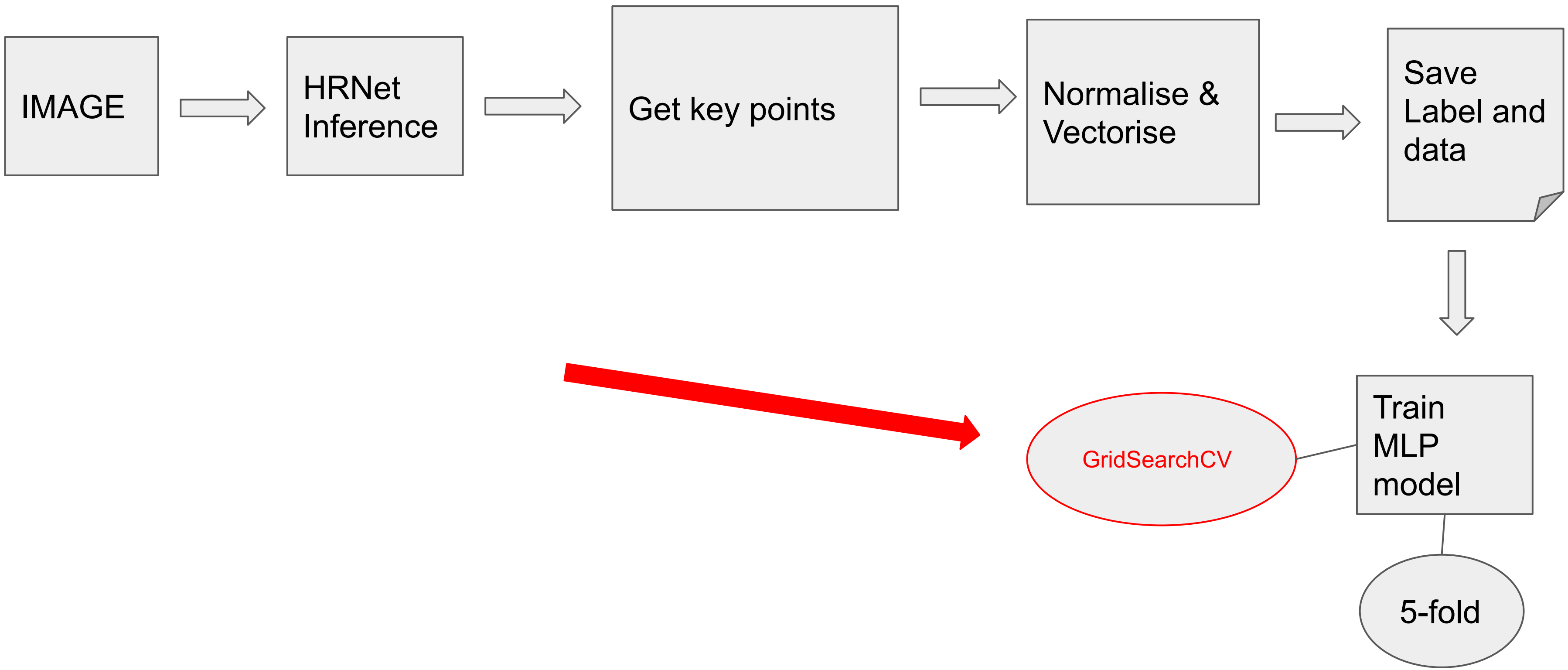


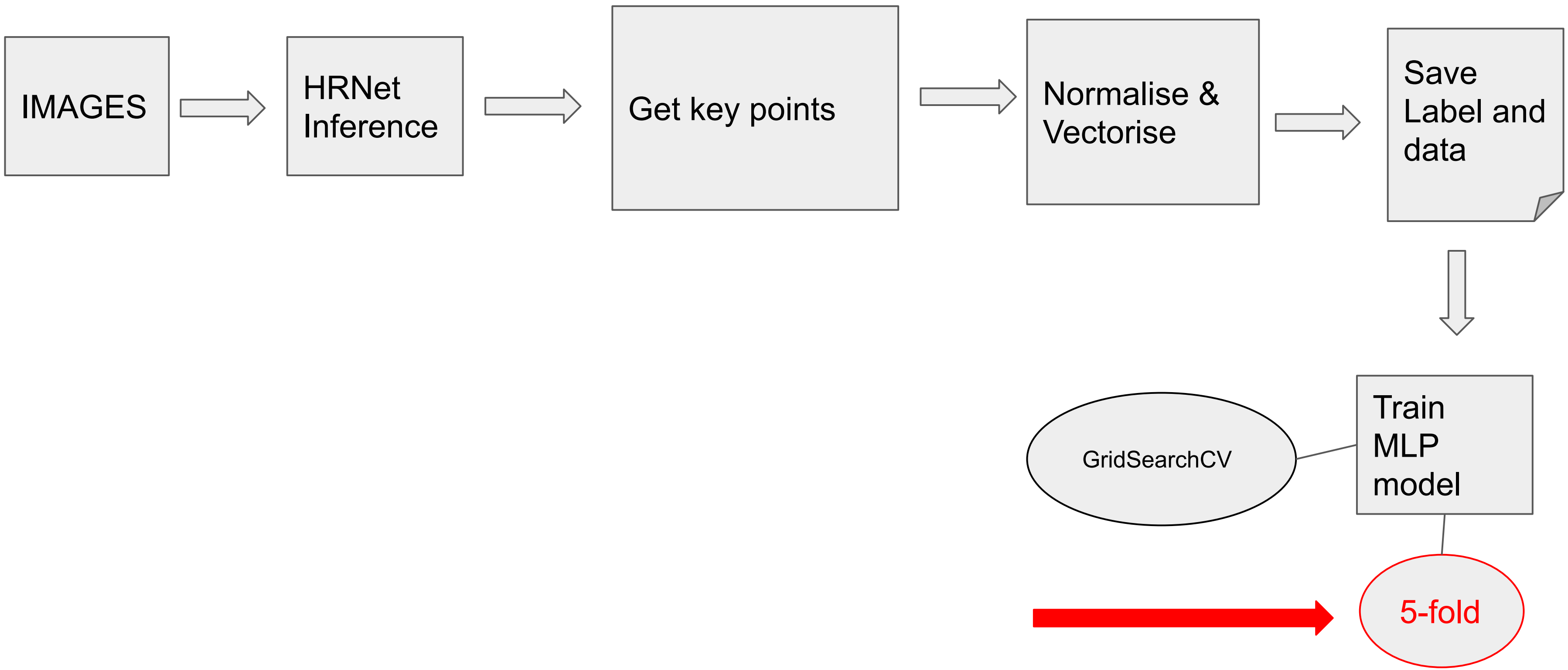
IMAGE
HRNet Inference
Get key points
Normalise & Vectorise
Save Label and data
Train MLP model
GridSearchCV
5-fold

GridSearchCV

- Module used to tune and find out the best combination of hyperparameters, giving us the best score for the MLP model

- Parameter options we tried :
    - 'learning_rate': ["constant", "invscaling", "adaptive"],
    - 'hidden_layer_sizes': [(30,30,30),(10,10,10)],
    - 'solver': ['sgd', 'adam'],
    - 'activation': ["relu", "logistic", "tanh"],
    - 'max_iter' : [10000]

- Best parameters found for our model :
    - **activation**: “tanh”
    - **hidden_layer_sizes**: “(30,30,30)”
    - **learning_rate**: “invscaling”
    - **max_iter**: “10000”
    - **solver**: “adam”


IMAGES
HRNet Inference
Get key points
Normalise & Vectorise
Save Label and data
Train MLP model
GridSearchCV
5-fold

5-fold Cross Validation

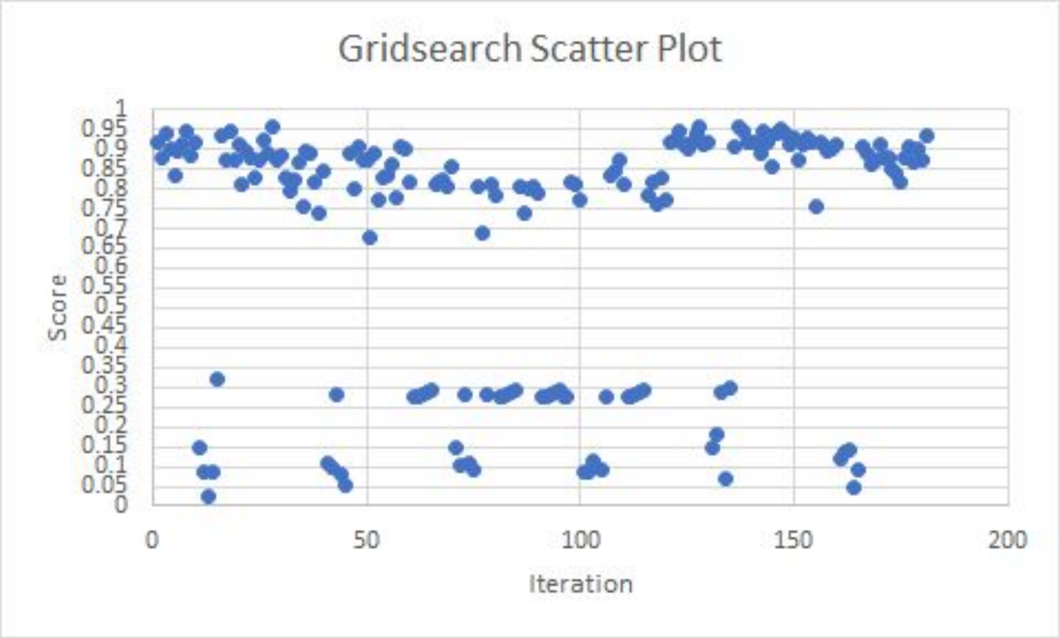
- Avoid overfitting of data
- Smoothen out any bias in data

In our implementation, **StratifiedKFold** has been used with *n_splits = 5*

Our Results on best model

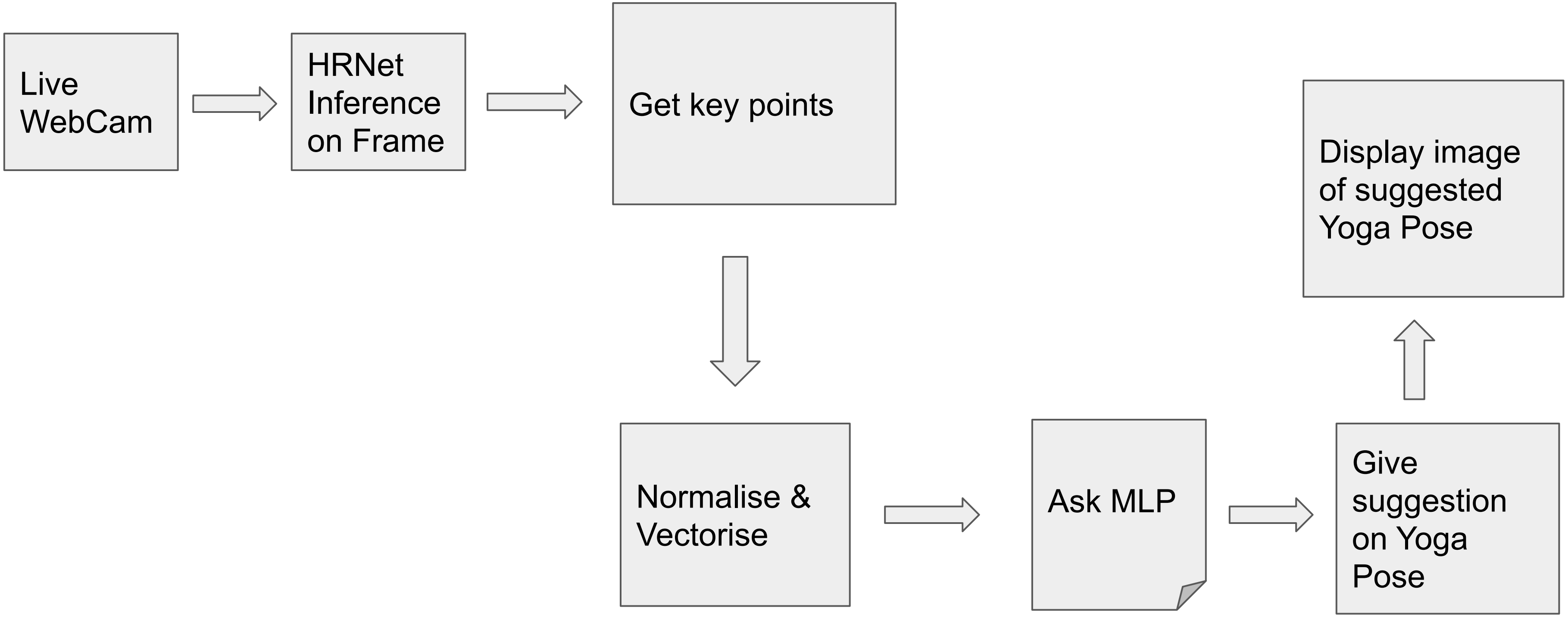
Worst Score: 0.055

Best score: 0.933 with fastest convergence

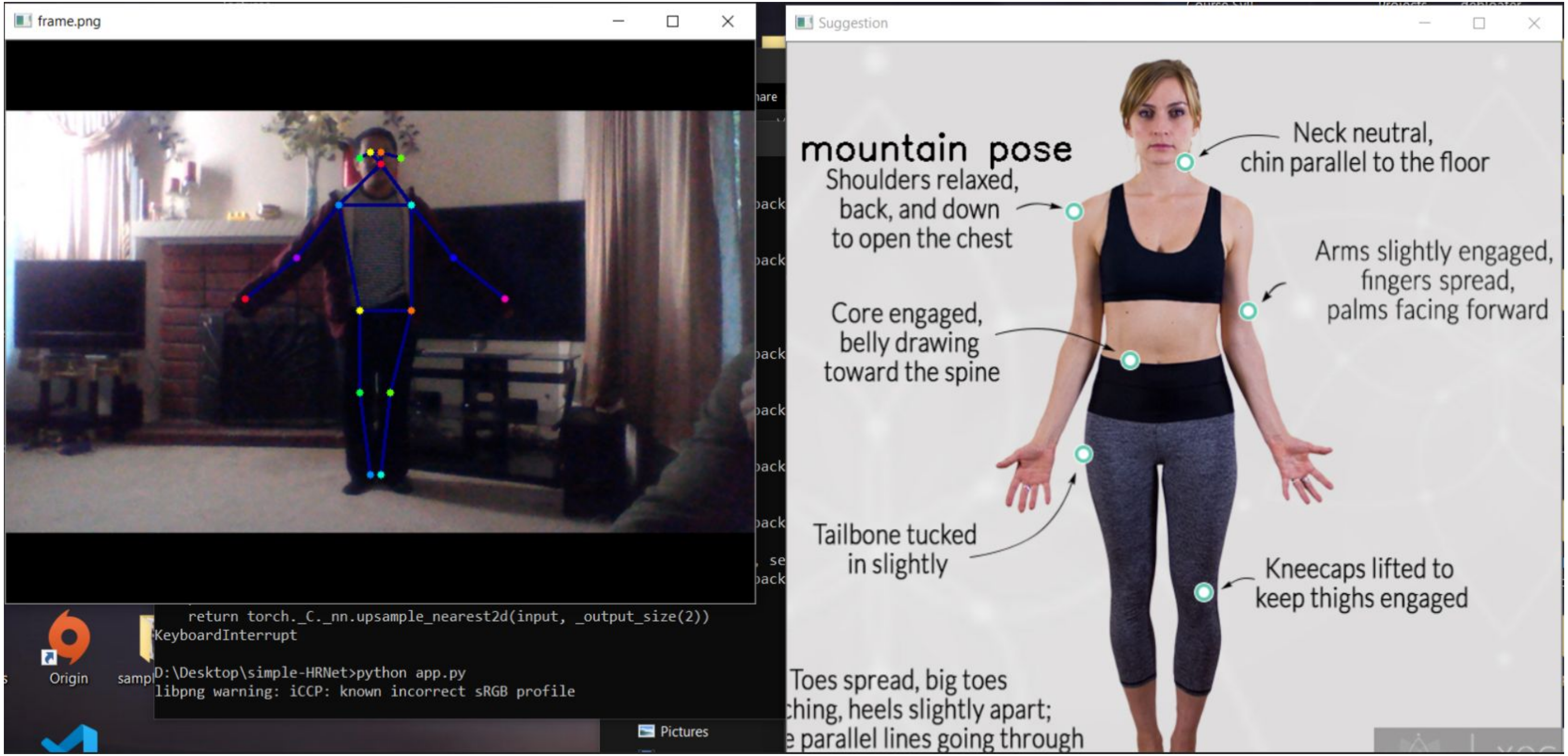


Architectural overview(Review):

Application Pipeline



Our Application



Demo

References

[1] Abu Hassan, Mohd Fadzil & Hussain, A & Md Saad, Mohamad Hanif. (2018). Polygonal Shape-based Features for Pose Recognition using Kernel-SVM. Journal of Telecommunication. 10.

[2] Lv, Fengjun, Nevatia, Ramakant Recognition and Segmentation of 3-D Human Action Using HMM and Multi-class AdaBoost. Computer Vision -- ECCV 2006

[3] Shahbudin, Shahrani. (2012). A Simplified Shock Graph for Human Posture Classification Using the Adaptive Neuro Fuzzy Inference System. Journal of Information and Computational Science. 9. 2035–2048.

[4] Zhihui Su, Ming Ye, Guohui Zhang, Lei Dai, Jianda Sheng. Cascade Feature Aggregation for Human Pose Estimation. arXiv:1902.07837v3, 2019

[5] Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, Everybody Dance Now. arXiv:1808.07371v2, 2019.

[6] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. arXiv preprint arXiv:1812.00324, 2018.

[7] Federica Bogo and Angjoo Kanazawa and Christoph Lassner and Peter Gehler and Javier Romero and Michael J. Black(2016): Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In: 14th European Conference on Computer Vision(2016).

[8] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P.,Schiele, B.: DeepCut: Joint subset partition and labeling for multi person pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 4929–4937 (2016)

[9] Dario Pavllo, Christoph Feichtenhofer, David Grangier, Michael Auli. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In CVPR 2018.

[10] Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee.: (2019) Camera Distance-aware Top-down Approach for 3D Multi-person Pose Estimation from a Single RGB Image. In ICCV 2019

[11] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In ICCV, 2017.

[12] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In ECCV, 2018.

[13] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for Human Pose Estimation. arXiv:1603.06937v2

[14] Our Code: https://github.com/bbdavidbb/YogAi

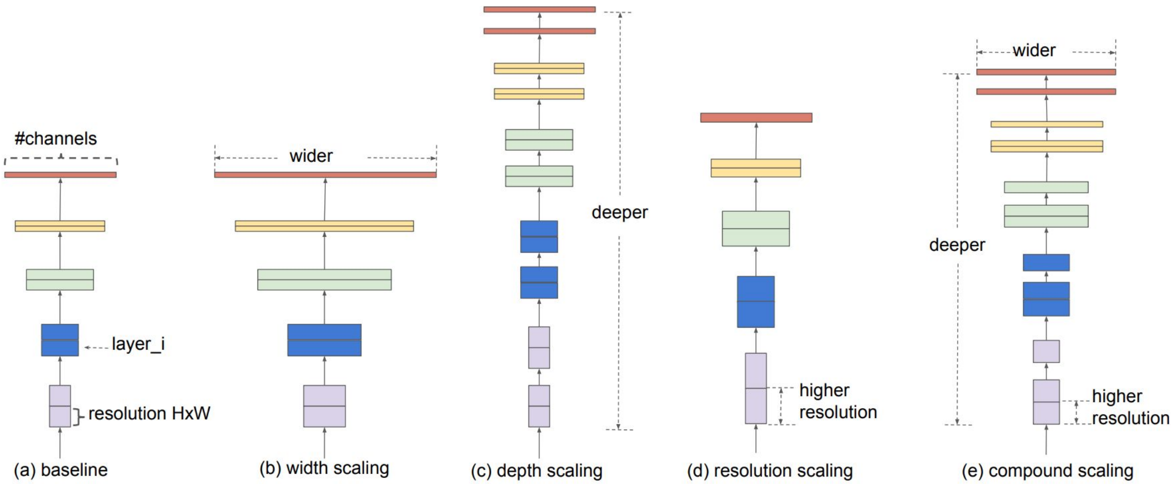
Bonus: Efficient Net

Aniket Chandak
Ashraf Saber
David Bui
Deanne Kshipra Charan
Hardik Kumar
Siddartha Thentu

Model Scaling

Scaling Methods:

- Depth
- Width
- Resolution



Model Scaling: https://arxiv.org/pdf/1905.11946.pdf

Model Scaling

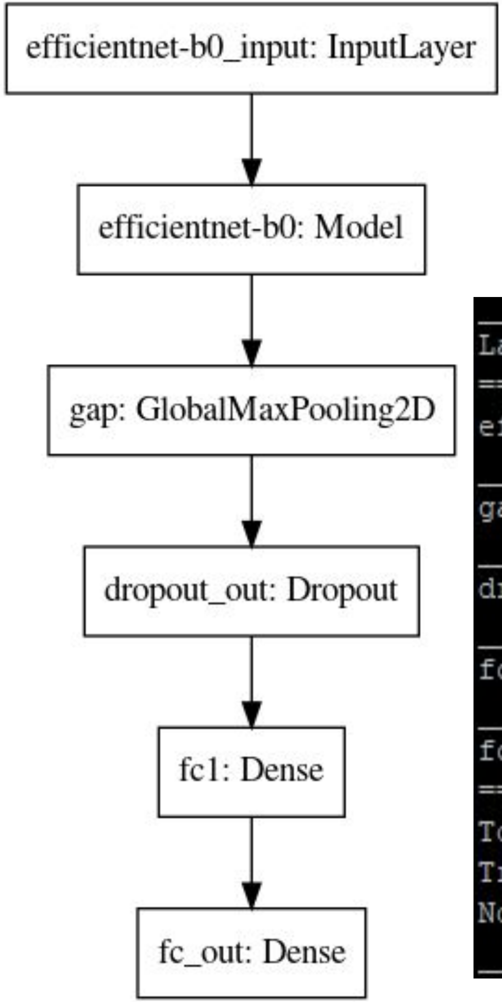
- The paper studies the impact of scaling models on performance.
- The paper proposes a **scaling method** that scales **all dimensions uniformly (width, depth, resolution)**.
- Constant coefficients α , β , γ are determined by a small grid search on the original model
- The scaling method works well on MobileNets and ResNets
- Effectiveness of scaling depends on the baseline model

Transfer Learning

Approach:

- Loaded pre trained Efficientnet B0
- Net(weights='imagenet', include_top=False, input_shape=input_shape)

```
Total layers before:
250
Total layers after:
246
Found 40000 images belonging to 100 classes.
Found 10000 images belonging to 100 classes.
Found 10000 images belonging to 100 classes.
```



- Add GMP ,
- Dropout(0.2),
- Dense(1024),
- Dense(100)

Freeze previous layers
Fine Tune a few layers.

Layer (type)	Output Shape	Param #
efficientnet-b0 (Model)	(None, 7, 7, 1280)	4049564
gap (GlobalMaxPooling2D)	(None, 1280)	0
dropout_out (Dropout)	(None, 1280)	0
fc1 (Dense)	(None, 1024)	1311744
fc_out (Dense)	(None, 100)	102500

Total params: 5,463,808
Trainable params: 1,456,260
Non-trainable params: 4,007,548

When fine tuning only the last two layers:

- Overfitting from 1st epoch
- Keras bug :: Cannot freeze Batch Normalization layers.

```
This is the number of trainable layers after freezing the conv base: 2
Epoch 1/20
313/313 [==============================] - 3236s 10s/step - loss: 6.5986 - acc: 0.1517 - val_loss: 4.8063 - val_acc: 0.1101
Epoch 2/20
313/313 [==============================] - 3227s 10s/step - loss: 4.2718 - acc: 0.3007 - val_loss: 4.6755 - val_acc: 0.1314
Epoch 3/20
313/313 [==============================] - 3240s 10s/step - loss: 3.6659 - acc: 0.3503 - val_loss: 4.6594 - val_acc: 0.1334
Epoch 4/20
```

Results

- Trained last 7 layers + All Batch Norm layers
- Upsampled images from 32x32 to 224x224
- Augmented training data using flip,rotate etc
- Optimizer=RMSProp lr=0.0064, decay=1e-6
- Batch_size = 128
- EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=5)
- Ran for 50 epochs but starts overfitting at 17 epochs.

```
This is the number of trainable layers after unfreezing the conv base: 102
```

```
312/312 [==============================] - 4081s 13s/step - loss: 1.3031 - acc: 0.6724 -
Epoch 16/50
312/312 [==============================] - 4085s 13s/step - loss: 1.2195 - acc: 0.6782 -
Epoch 17/50
312/312 [==============================] - 4086s 13s/step - loss: 1.2531 - acc: 0.6810 -
Epoch 00017: early stopping
79/79 [==============================] - 121s 2s/step - loss: 1.4498 - acc: 0.6475
acuracy =  0.6475
```

Conclusion

- State of the art is 88%

How to achieve?

- Fine tune all layers
- Decrease the learning rate more
- Add more regularization effects
- Increase the training set

References

[1] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." arXiv preprint arXiv:1905.11946 (2019).