Posture Recognition and Keypoint Detection in Smart Beds with Deep Learning

Orkhan Bayramli[†]

Abstract—**This project report deals with two types of Computer Vision problems namely Pose Classification and Keypoint detection on the publicly available pressure map dataset. Convolutional Neural Networks have been used to solve these problems. The dataset has been extracted from the text files and saved as images to form a suitable type of dataset for CNN models. Apart from CNNs, K-means machine learning algorithm was used to identify the clusters formed in both raw dataset and encodings extracted from the model. The encodings and the raw dataset have been reduced to two-dimensional space by the help of PCA and t-SNE reduction algorithms. The goal of the project is to classify the sleeping postures and assess a health metric.**

Index Terms—**Convolutional Neural Networks, Unsupervised Learning, Image Classification, Keypoint Detection, Image Processing.**

I. Introduction

Sleep is a natural activity, which is essential for life and physical well-being. It has been shown that sleeping affects symptoms of many diseases and plays an important role in systemic physiology, including metabolism and the functions of the immune, hormonal, and cardiovascular systems [1]. As such, recent studies have shown in-bed postures have a high influence on occurrence of sleep diseases such as apnea, pressure ulcers, and even carpal tunnel syndrome [2]. For instance, sleep apnea patients are recommended to only sleep in the side-positions [3].

For that sake, Deep Learning especially the Convolutional layers can be quite helpful identifying the sleeping postures thanks to their powerful characteristics of detecting the high level features from the images. In addition, the same characteristics can also be used to detect and localize the keypoints which will help us to assess some metrics for the quality of sleeping posture.

In this paper, I tackle with two types of Computer Vision problems namely *Pose Classification* and *Keypoint Detection* to identify and assess healthy sleeping postures. The dataset used in this project is a publicly available pressure map dataset [4] which contains different sleeping positions of subjects recorded via the sensors embedded to the mattresses. The first problem has been solved by Davoodnia et al. in the following paper [5] with Convolutional Neural Networks. I have re-implemented this approach in TensorFlow and later applied some Machine Learning algorithms and Dimensionality Reduction techniques such as K-means, PCA, and t-SNE to analyze the encodings. On the other hand, the second problem is a new idea which was conducted by labelling the dataset and predicting the degree of a triangle formed by three keypoints to detect how curved the subject's spine is.

Summary of contributions the summarized list of contributions is as follows:

- **Pose Classification:** Re-implementation of Pose Classification problem from Davoodnia et al. [5] in TensorFlow and Python.
- **Dimensionality Reduction:** Application of Dimensionality Reduction algorithms such as PCA and t-SNE to 2-Dimensional space. Both raw data and encodings extracted from the model were reduced to 2-Dimensional space.
- **Visualization:** The visualization of the reduced data with corresponding images to have a better understanding of possible clusters.
- **Keypoint Detection:** Predicting the degree of a triangle formed by three keypoints to detect how curved the subject's spine is.

This report is structured as follows. The system and data models are respectively presented in Sections II and III. The learning framework is in Section IV and its performance evaluation is carried out in Section V. Concluding remarks are provided in Section VI.

II. Processing Pipeline

A. Classification

I have used two deep learning models to identify the poses in the dataset. The first is a feed forward neural network while the second containing convolutional layers.

Feed-forward Network ***(FFN)***: The overall structure of the feed forward network is depicted in Fig. 1. In general, it has four layer namely Rescaling, Flatten, and two Dense layers one of them being the output layer. The input images are of shape (256, 128, 1) and that's why the rescaling layer has the same shape. It is there to divide the input pixel values by 255 which is the maximum value that a pixel value stored in 8-byte can take. Since the neural networks work better with smaller values, rescaling layer eases off their job. Later, the Flatten layer flattens out the input pixels into a vector having a dimension of 32768 which is the result of 256 * 128 * 1. Then, everything is fed up to the hidden layer with the dimension of 128 with ReLU activation function. Lastly, the output layer tries to predict which output node the input image falls into by giving us the probability from its softmax activation function.

Convolutional Neural Network ***(CNN)***: Convolutional Neural Network Fig. 2 model consists convolutional layers

[†]Department of Information Engineering, University of Padova, email: {orkhan.bayramli}@studenti.unipd.it

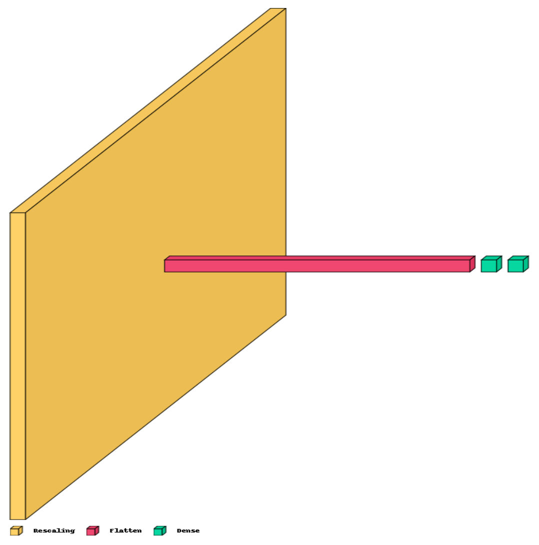


Fig. 1: Shallow Feed Forward Model for classification

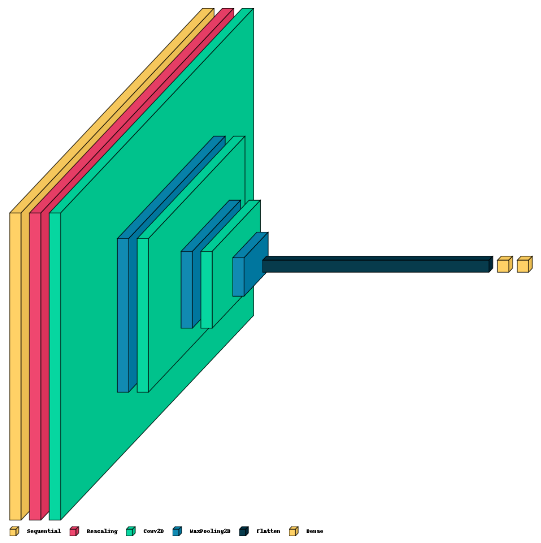


Fig. 2: CNN Model for classification

that try to learn the low and high level features from the images, in our case, sleeping postures.

- **Sequential layer**: The first layer is a Sequential Layer that comes from Keras API and helps us to group the layers together. Indeed, the whole network is an instance of Sequential layer. The sequential layer inside of the model constitutes for Data Augmentation. I have embedded the data augmentation logic inside of the network which is turned off during the evaluation and prediction modes. Embedding the data augmentation inside of the network helps us to benefit from Graphical Processing Unit (GPU) while augmenting the images and speeds up the training process. The sequential layer consists of four data augmentation layers namely:
  - **RandomRotation**: ±5 degrees of rotation.
  - **RandomZoom**: zoom in and out up to 10 percent.
  - **RandomBrightness**: brightening up to ±5 percent.

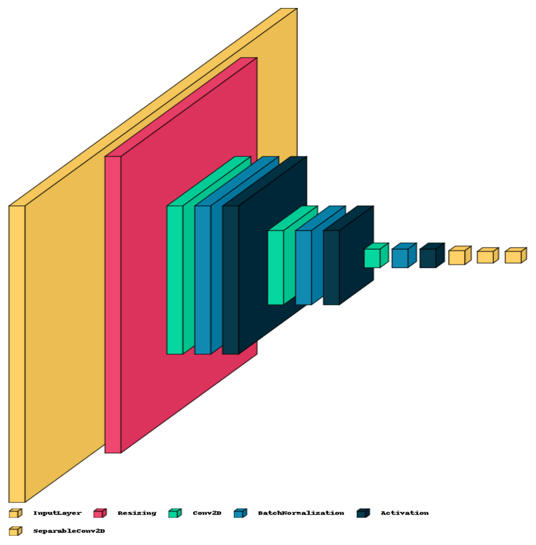


Fig. 3: CNN Model for keypoint detection

    Negative values means darkening.
  - **RandomContrast**: contrasting up to ±5 percent.
- **Rescaling Layer**: Rescaling the pixel value in the range of [0, 255].
- **Convolutional Layer**: The model architecture contains 3 different convolutional layers. The shape of filters used across the convolutional layers are same which is (3 x 3). The number of filters are 16, 32, and 64 in an increasing order. "same" type of padding results in padding with zeros evenly to the left/right or up/down of the input. When padding="same" and strides=1, the output has the same size as the input which is the case in our network. ReLU activation function was used in the convolutional layers.
- **MaxPooling Layer**: MaxPooling downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window for each channel of the input. The default (2 x 2) pool size has been used in the MaxPooling layers. In general, there are 3 MaxPooling layer each of them following a Convolutional Layer. In our case, the MaxPooling layer with "valid" padding halves the spatial dimensions of the input signal by the factor of two.
- **Flatten Layer**: Flatten Layer was used right after the last MaxPooling layer, flattening out the three dimensional input signal into one.
- **Dense Layer**: The last two layers of the network are of Dense Layer type. The first Dense layer has 128 nodes which acts like encodings learned from the images with the activation function being ReLU. The second Dense Layer has three nodes which correspond to the number of categories in our dataset namely left, right, and supine sleeping postures. To get the probabilities, softmax activation function was used in this layer. The second Dense Layer is also the output layer of our model.

B. Keypoint Detection

For the keypoint detection type of computer vision problem, the model Fig. 3 that I'm using the convolutional layers but the output layer is what makes it different than the previous classification problem. Since the dataset has three keypoint tuples of x and y coordinates, the output layer will have the shape of (1x1x6). 6 stands for 3 keypoints multiplied by 2 coordinates. Different than the CNN model used in the previous classification problem, this network utilizes *Batch-Normalization* and *SeperableConv2D* layers.

- **Convolutional Layer:** There are three *normal* convolutional layers in the model. The output filters are as following in descending order: 64, 32, 32. "same" type of padding was used in each convolutional layer. On the other hand, the strides are as following: 2, 2, 4. I have adjusted the strides and filter sizes to give me the shape of (1x1x6) layer at the end of the model.
- **BatchNormalization:** I have read some blogs about the usefulness of batch normalization in neural. Out of curiosity, I decided to include it inside of the model.
- **Activation:** The activation function used for convolutional layers is the ReLU. I read some articles about where to place the activation and decided to put it right after BatchNormalization layer which inded comes after Convolutional layer. However, the last activation function placed after the last layer of the model is sigmoid since I need the values between 0 and 1 which will be multiplied by 255 later to get the coordinates.
- **SeperableConv2D:** Seperable convolutional layers are slightly different than the *normal* convolutional layers. Separable convolutions consist of first performing a depthwise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes the resulting output channels. The output number of filters correspond to the number of keypoints and I have used the combination of three seperable convolutional layers with the strides being 1, 1, and 2 to get the final shape of (1x1x6).

III. Signals and Features

A. Measurement setup

The dataset used in this project contains a set of measurements made by pressure sensors installed on several mattresses with the purpose of collecting in-bed posture data. The data collection was performed by *Pouyan et. al (2017)* [4]. The dataset is publicly available and hosted by the Physionet databank here. The data was collected using Force Sensitive Application (FSA) pressure mapping mattresses with a sampling frequency of 1Hz. This pressure mat is light, thin, and flexible enough so it can cover the entire mattress. The sensors were uniformly distributed across a 32 x 64 mat with each sensor being almost one inch apart. 13 subjects participated in the experiment and performed eight standard bed postures (and some slight variations, summing up to 17 unique postures), which were summarized (labeled) as five common bed postures: *left, right, left fetus, right fetus, and supine*. Each subject performed each pose for at most two minutes.

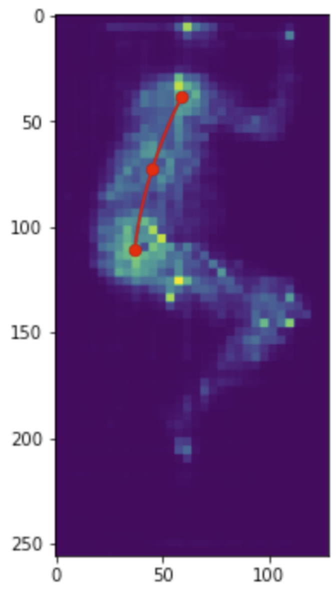


Fig. 4: A sample from the keypoint detection dataset.

I have used two datasets for the two different types of problems: one for classification and another for keypoint detection.

B. Pre-processing

- **Image Classification**: The images were stored in text files. I wrote a script to extract the images from the text file and interpolate them to 256 x 128 matrix with area method available in the image API of TensorFlow. In total, I have kept only three sleeping postures namely left, right, and supine. I have taken the right fetus and left fetus positions as left and right. In total, I had around 2056 data samples. After generating the images, I manually checked the images to detect the corrupt ones. Then, I splitted the dataset into train and test dataset with the ratio of 0.9. Later, when I'm reading the images from disk, I splitted the train dataset into train and validation ones. I took the 20 percent of training samples as validation. The final numbers are as following: 1480 for trainig, 369 for validation, 205 for testing. I used the validation dataset for model selection and to prevent overfitting.
- **Keypoint Detection**: I have taken the subset of the image classification dataset for keypoint detection. I had around 350 data samples for this task. I labelled all the images using the the software called "LabelMe". An example from the keypoint detection dataset is shown in Fig. 4. In total, there are three keypoints: **shoulder, mid-body, hip**. Then, I implemented a function to detect the degree of a triangle formed by these three keypoints which is meant to asses some sort of metric for healthy sleeping posture. More the angle is close to the straight, better the sleeping posture is. The curved lines between the keypoints were

drawn by using the Krogh's interpolation from the Scipy library. Lastly, I built a Python script to generate a JSON file containing all the relative paths to the images and their corresponding keypoint coordinates which was ,in turn, needed for the model's custom Dataset class.

IV. Learning Framework

A. Classification

The loss function used for the classification is Adam [6] with the default learning of 1e-3. I was using the "Reduce Learning Rate on Plateau" callback provided in Keras to decrease the learning rate by the factor of 0.1. Another callback was the "Early Stopping" to stop the training if the validation accuracy did not improve anymore. The loss function was the Sparse Categorical Crossentropy which calculates the loss on the output of model with the labels being 0, 1, 2 corresponding to the sleeping postures. The batch size was 6. The training took around 15 epochs. After the training, I visualized the convolutional layers to check what parts of images they tend to learn, or to see if they learn at all.

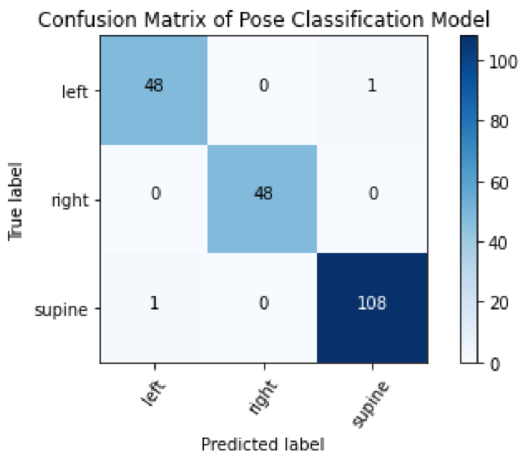
B. Keypoint Detection

Although the types of problems are different, I used the same hyperparameter values for also the keypoint detection model which indeed is a convolutional model. The only notable difference was in the loss function which is "Mean Squared Error" for this model. It tries to calculate how far off the predicted keypoints are from the ground-truth keypoints. It took around 23 epochs to finish training.

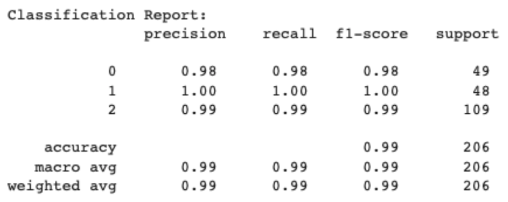
V. Results

For the pose classification, I have used the "Classification Report" and "Confusion Matrix" from the scikit-learn library. The results are depicted in the following Fig. 5. Since the images are rather simple and having less number of pixels, our classification model could learn them pretty well. The results are performed on the test dataset. Note: The test and train datasets were coming from the same distribution. When I was writing the report, I thought it would be better idea to use the images performed by certain subjects only for testing and the rest for training.

In the Fig. 6 we can see the predicted keypoints from the keypoint detection model. Model seems to be learning the spatial locations of keypoints pretty well. One issue could be when I was labelling the dataset, in the beginning, I was not quite sure where exactly put the keypoints. I just followed my common-sense. The results may not be too accurate because of that but if we take into account that only the angle values below certain number like 160°are considered as bad sleeping posture, the results are acceptable.



(a) Confusion Matrix.



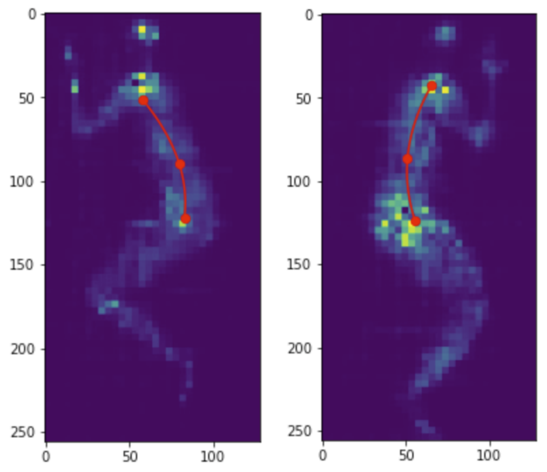
```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        49
           1       1.00      1.00      1.00        48
           2       0.99      0.99      0.99       109

    accuracy                           0.99       206
   macro avg       0.99      0.99      0.99       206
weighted avg       0.99      0.99      0.99       206
```

(b) Classification Report

Fig. 5: The results of the classification model.



(a) Keypoint prediction 1. (b) Keypoint prediction 2.

Fig. 6: Keypoint predictions.

VI. Concluding Remarks

In conclusion, it was shown that it is possible to detect the dominant sleeping postures with high accuracy. Moreover, it is also possible to get a metric to assess a healthy sleeping posture. Applicability of project extends to the hospitals where mostly the unsupervised monitoring is prevalent because of the high number of beds. Another application area is in the Smart Home environment as a health assessment tool. I believe there is a lot of room for further development in the keypoint detection as it maybe also possible to detect the angles in the different junctions of human body where required. While doing the project, I learned how to build custom models and dataset in Keras. I also gained knowledge in Image Processing. More importantly, I was able to work independently and apply my ideas to the project. The most difficult part for me was to build the network from scratch, make sure all the shapes are correct, and the model is working.

References

[1] C. C. Panel, N. F. Watson, M. S. Badr, G. Belenky, D. L. Bliwise, O. M. Buxton, D. Buysse, D. F. Dinges, J. Gangwisch, M. A. Grandner, K. Clete, R. K. Malhotra, J. L. Martin, S. R. Patel, S. F. Quan, and E. Tasali, “Joint consensus statement of the american academy of sleep medicine and sleep research society on the recommended amount of sleep for a healthy adult: methodology and discussion,” in *Sleep*, vol. 38, pp. 1161–1183, 2015.

[2] S. J. McCabe, A. Gupta, D. E. Tate, and J. Myers, “Preferred sleep position on the side is associated with carpal tunnel syndrome,” in *Hand*, vol. 6, pp. 132–137, 2011.

[3] R. D. Cartwright, “Effect of sleep position on sleep apnea severity,” in *Sleep*, vol. 7, pp. 110–114, 1984.

[4] M. B. Pouyan, J. Birjandtalab, M. Heydarzadeh, M. Nourani, and S. Ostadabbas, “A pressure map dataset for posture and subject analytics,” in *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pp. 65–68, 2017.

[5] V. Davoodnia and A. Etemad, “Identity and posture recognition in smart beds with deep multitask learning,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3054–3059, 2019.

[6] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.