THREE DIMENSIONAL POSE ESTIMATION FROM MONOCULAR IMAGE USING DEEP NEURAL NETWORK

[1]Denilson

[1]JL. TK Al Kindi No. 126 Rt 004/001 Kel. Cipayung Jaya Kec. Cipayung Depok (denilson020898@gmail.com)

[2]Dr. Dharmayanti, ST., MMSI.

[2]Jl. Samiaji VIII/336 Rt 007/019 Kel. Sukmajaya Kec. Mekar Jaya Depok II Tengah (dharmayanti77@gmail.com)
Informatics Engineering, Faculty of Industrial Technology,
Gunadarma University

ABSTRACT

Digital technologies have been developed rapidly in application and science may produce digital track records that are actually useful. Digital data are available in a huge number and are predicted to increase. One way to utilize this data is to create a mapping function that finds a correlation between domains from the data itself as a reference. Digital data in form of sequence of images or videos are latent which mean data itself has some hidden semantic meanings. This research is about making a mapping function that maps two dimensional images into three dimensional human pose keypoints using deep neural network modeling. The software is built in steps that involve data preprocessing, model architecture design, self-traning deep neural network, and visualization. The model consists of some blocks of residual networks that sum up its inputs and outputs. The result from testing explains that the theories and data are correct and runs correctly using new data as input.

Keywords: Artificial Neural Network, Computer Vision, Deep Learning, Monocular Image, Pose Estimation

INTRODUCTION

The use of computerized technology by humans always leaves traces that are stored in the form of digital data. This track record is evidence of human behavior and characteristics so that it is used as a reference for technology and science in the future. Digital data that is generally utilized by humans includes text, audio images, visual images, and audio-visual images stored in a storage medium. The large amount of data available and it is predicted that it will increase, makes the human lifestyle always be in digital technology.

Deep learning modeling can map one domain to another independently using deep neural network. Deep learning can be done by self-computation which is very dependent on the quantity and quality of good data. The study of using artificial neural networks can be used to develop technology, especially in the field of computer vision, such as estimating the three-dimensional poses of the human body contained in a monocular image.

RESEARCH

This study discusses the use of image data as a reference in learning and implementing

deep neural network models to locate and map the three-dimensional coordinates of human body poses in a series of images locally. The pose used is not grounded, which means that the coordinates of the pose are not centered on a certain floor point. Processing the application prioritizes two important steps which include data processing and modeling. The application made can display a three-dimensional graph plot resembling the anatomical structure of the human body in accordance with the estimated poses of the input image. The results of the learning of the model are displayed in a two-dimensional graph for further analysis.

The deep neural network model is implemented using the PyTorch framework. The two datasets used are processed first so that they meet PyTorch's requirements for deep learning. Each model is then used against the application inference dataset. The estimation process and results are further described in the form of a visual graphic.

STUDY

Data Analysis

Modeling data used is mapping data from two-dimensional poses to three-dimensional poses. Two-dimensional poses are samples, while three-dimensional poses are targets. The data source is Human3.6M. Dataset is about information about human pose movement that stores images from several sides along with two-dimensional and three-dimensional poses (Ionescu et al, 2014).

Table 1. Learning Data for Model

File Name	Contents
rcams.pt	Camera position matrices
stat_2d.pt	Mean, std, and pose 2D
stat_3d.pt	Mean, std, and pose 3D
test_2d.pt	Pose 2D data for validation
test_3d.pt	Pose 3D data for validation
train_2d.pt	Pose 2D data for training
train_3d.pt	Pose 2D data for training

Data Pre-Processing

The Dataset and DataLoader classes in the PyTorch framework have the functionality to read and share data series stochastically. The DataLoader class can transfer information in the Dataset class to VRAM on the GPU in the form of a mini-batch so that it can be processed stochastically and parallel. The first step is to check whether the desired data is training data or validation data. DataLoader receives the object then performs data randomization and mini-batch sharing. The results of the object can be iterated to perform model training (Kingma et al. 2014).

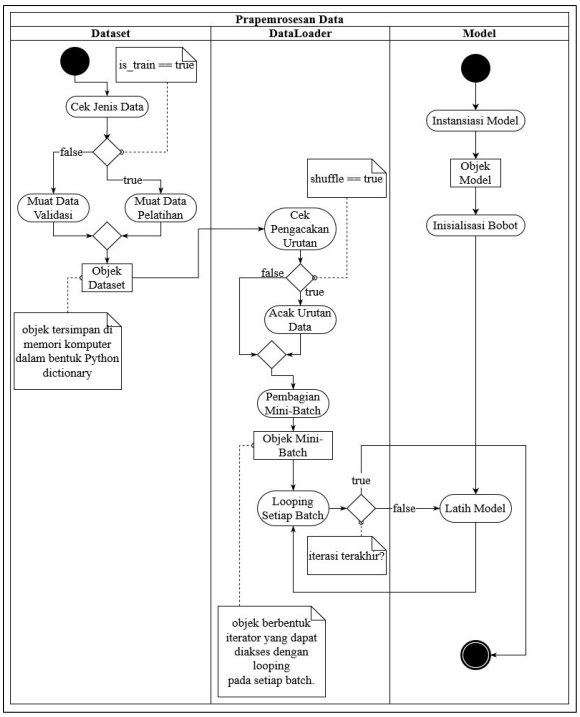

Figure 1. Data Pre-Processing Scheme

Model Architecture

The artificial neural network model architecture used has vector-shaped input with a size of thirty-two and vector-shaped output with a size of forty-eight. A series of layers connecting input and output is a residual network layer. The weight of each layer is randomly initialized with a normal distribution. A residual network layer is a network with a small and simple architecture that can be installed or dismantled in a modular fashion called ResLinear. The ResLinear layer performs the addition operation between input and output. A ResLinear has two linear layers, two Batchnorm layers, two Dropout layers, and two ReLU layers. The constituent components of a ResLinear layer with input size a and output size b (Martines et al. 2017).

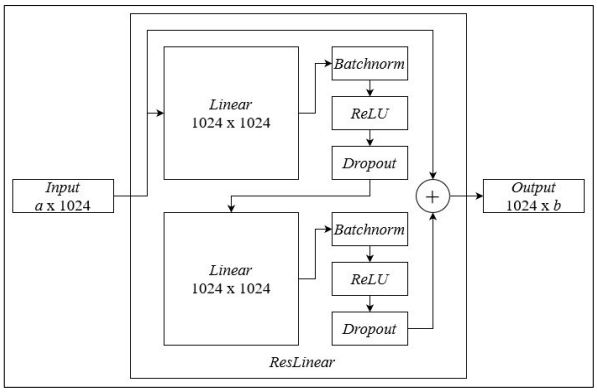

Figure 2. Modle Architecture

Model Training and Learning

Model learning is a continuation of data pre-processing. A new model is instantiated to produce a model object. The model object then initializes the parameter weights with random numbers from the normal distribution. The input batch originating from the DataLoader interaction is processed by the model using the feed-forward method. The provisional prediction results from the model are obtained which are then compared to the level of truth using the error function. The mean squared error function returns a number of the error quantity. This figure is a measure of how accurate the model's ability to produce relevant outputs. If the model is not in the "is_train" state, the error quantity obtained is stored as an array for analysis purposes. If the model is in "is_train" state, the model performs backpropagation to generate a weight gradient. The learning rate is then divided by two so that it becomes smaller. The weight adjustment is done by adding the weights with the results of the multiplication operation between the learning rate and the weight gradient. The updated model weights are stored along with the error quantity. The same algorithm will be repeated for each batch of inputs.

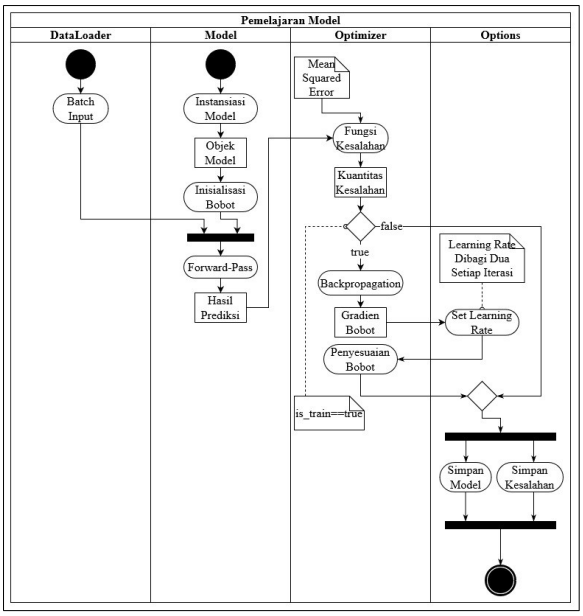

Figure 3. Model Learning Scheme

Inference Data Pre-Processing

The process of two-dimensional pose inference becomes faster when the operated two-dimensional image has a small resolution. The level of resolution used when recording the pose is 640 x 360. Reducing the size of the resolution must also be viewed in terms of the resulting image quality. Images also have a lot of static information such as the black pixel areas to the left and right of the image. The 290 x 290 resolution with the center point at the waist lock point is considered appropriate because it is able to reach all body poses and the image quality is still good. More efficient inference can be achieved by reducing the resolution and area of dead pixels.

Figure 4. Inference Data Pre-Processing

OpenPose

OpenPose is a two-dimensional human body pose estimation application. OpenPose accepts image input then looks for the key points of the two-dimensional pose. Two-dimensional pose key points are at local coordinates according to the image plane. OpenPose generates a "json" file which contains a hierarchy of poses according to the COCO-MS specification. The COCO-MS annotation contains eighteen key points of the human body in a specific sequence (Cao et al. 2019).

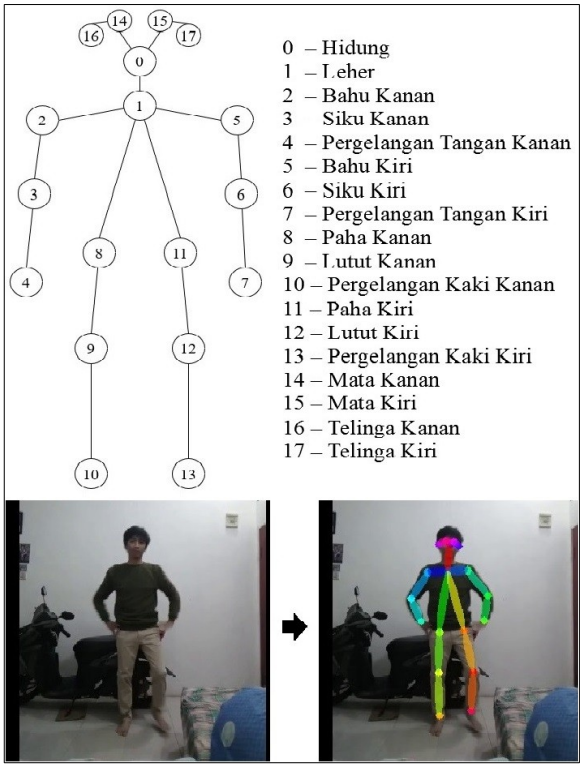

Figure 5. COCO-MS Keypoints

Model Inference

The set of key points generated by OpenPose has a COCO-MS specification that is different from the specification of the modeling dataset. The COCO-MS specification which has eighteen key points is converted into fifteen key points by joining the key points of both eyes and ears and creating a waist lock point based on the

average of the upper legs. This conversion is done so that the model can inference to the key points that represent the pose. Visualization of COCO-MS key point conversion using blue representing the right limb together with red representing the middle and left limbs.

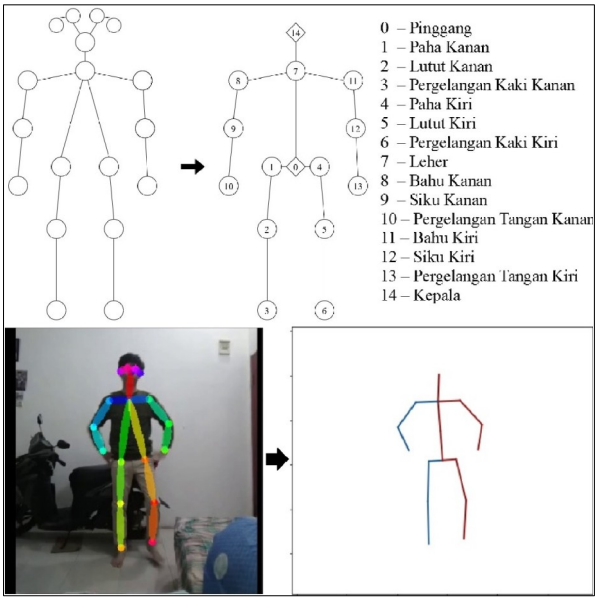

Figure 6. 2D Keypoints Conversion

Inference in the deep neural network model receives a two-dimensional pose key point that has been converted as input, then produces a three-dimensional pose key point as output. Model inference produces seventeen points that are in a three-dimensional plane. The three-dimensional pose key points have two new points covering the torso and nose. The torso point represents the curvature of the body in a three-dimensional pose so that the pose looks accurate. The nose point represents the direction towards the head. These two additional points clarify the orientation of the three-dimensional pose.

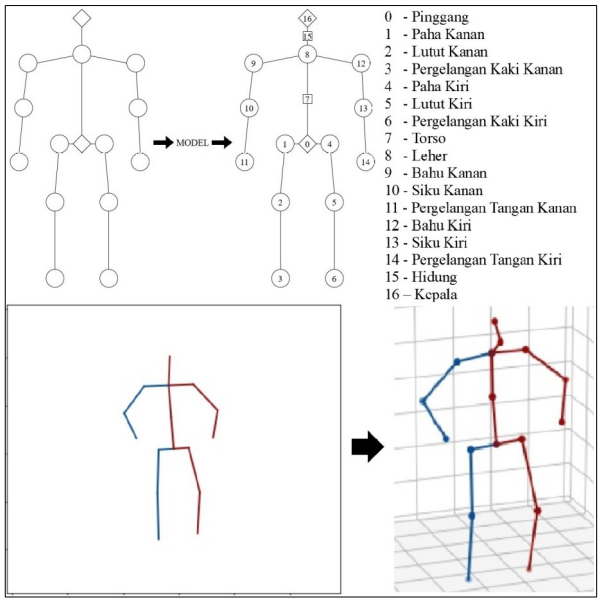

Figure 7. Model Inference

Visualization

Visualization includes the use of the application in one frame as a whole. There are four figures, each of which represents the steps in the experimental stage. The first figure (top left) contains inference data preprocessing where the image resolution is changed to 290 x 290. The second figure (top right) depicts the two-dimensional pose obtained by OpenPose. The third figure (bottom left) illustrates the conversion of OpenPose key points with the COCO-MS specification to model-matched key points. The fourth figure (bottom right) depicts the three-dimensional pose generated by the model into a three-dimensional coordinate system.

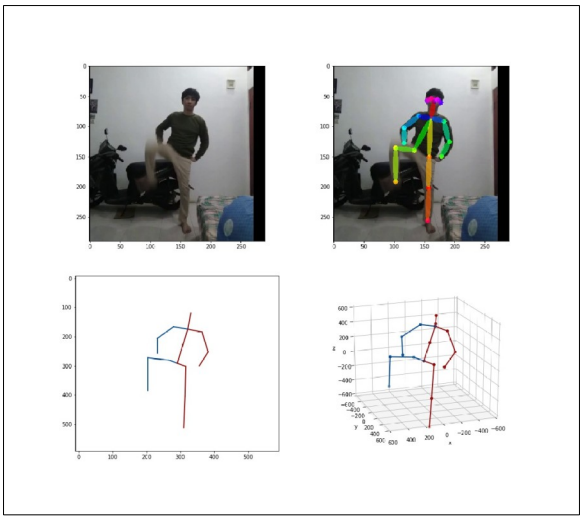

Figure 8. Application Visualization

Python Scripting

The steps taken at the production stage are still separated in several cells in the Jupyter Lab interactive development environment. The cells are then combined into a script called "run.py". This is done so that application users can run the application easily. Applications can be run by entering the command "python run.py" provided all python version 3 dependencies have been installed.

Model Learning Outcomes

Studying the model for ten epochs shows that the model error in estimating three-dimensional poses decreases with each epoch. The learning rate which is halved in each epoch affects the learning of the model where the model adjustment is more accurate. Adaptation of model weight occurs drastically at epoch0 to epoch3. Epoch4 and so on uses a learning rate that is getting smaller so that the model is getting more careful in adapting.

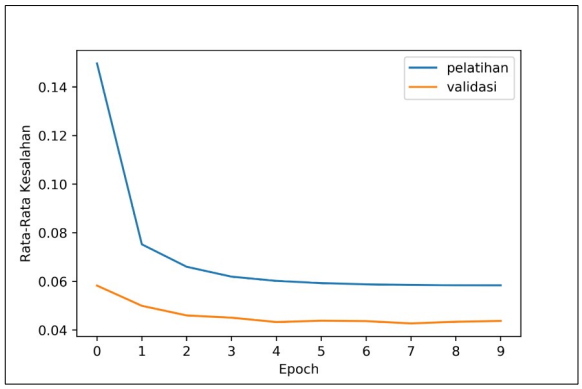

Figure 9. Model Learning Outcomes

Application Trial

Good inference will occur if the steps in the trial stage do not experience errors. The image quality and the flawless poses also influence the process from start to finish. Image preprocessing on precise inference data makes it easier for OpenPose to find the complete two-dimensional pose key points. Complete OpenPose key points then qualify for conversion to the desired specification. The information is then passed on to the model to get the key points of the three-dimensional pose.

Defective pose quality results in flawed three-dimensional pose estimates. The occlusion of a pose in a two-dimensional image can remove a part of the body. Missing this part of the image causes errors in later steps. The lock point of the right arm is lost when the arm is pointing straight towards the camera lens, causing occlusion. This causes OpenPose to be unable to find the right arm lock point and assigns zero value to it. The three-dimensional key point conversion and inference processes also result in unrealistic poses.

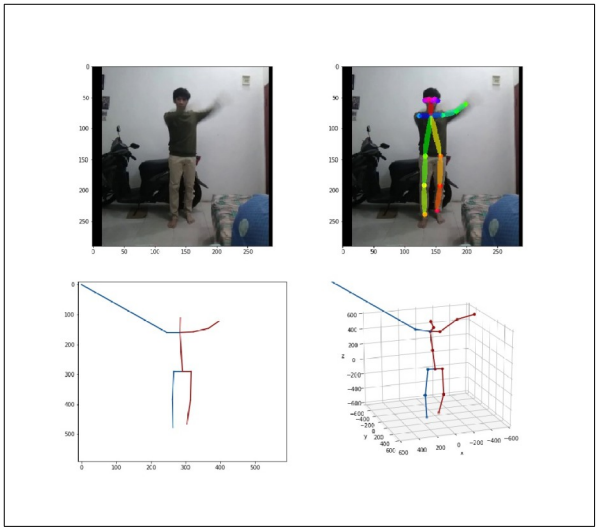

Figure 10. Missing Pose Inference

Errors can also occur in the key point inference process. If OpenPose issues an ambiguous output where there are key points that are considered part of the human body. OpenPose generates a double key point that does not match the required specifications despite successfully detecting the pose completely. The output that does not match the model specifications results in a broken three-dimensional pose estimate.

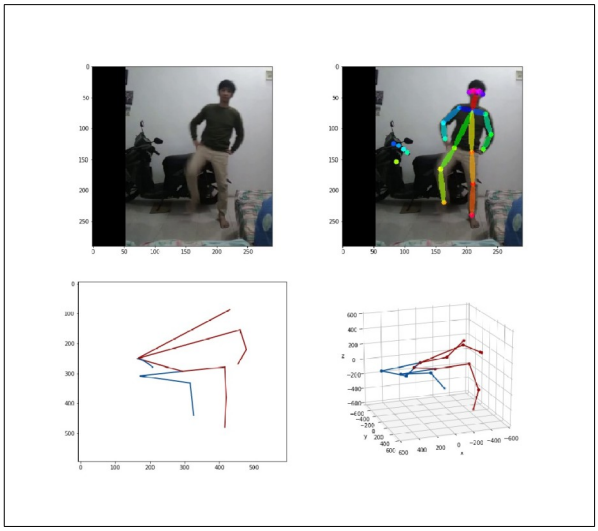

Figure 11. Error Pose Detection

CLOSING

The application of estimating three-dimensional poses using a deep neural network model has been successfully trained. This model performs independent learning using two-dimensional pose information as input and three-dimensional poses as output. The model is tested to find three-dimensional poses in the inference data. The model studied for ten epochs with an average final error value of 0.0584 on the training data and 0.0437 on the validation data. The error value of the training data is still greater than the value of the validation data. This indicates that the model is still in an underfitting condition where the difference between the two values is relatively large. A better model can be obtained by training the model with a greater number of epochs and stopping when overfitting begins.

The development of this deep neural network model still uses a minimalist architecture, data with one domain, and has inefficient stages. Further development is recommended to use a more efficient architecture. The residual network architecture is the best architecture at the time of this writing. More efficient learning algorithms are also suggested in future studies. Data with a wider domain is also of importance such as estimating poses in certain animals.

BIBLIOGRAPHY

[1] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., dan Sheikh, Y. A. (2019).Openpose: Realtime multi-person 2d pose estimation using part affinity fields.IEEE Transactions on Pattern Analysis and Machine Intelligence.

[2] Ionescu, C., Papava, D., Olaru, V., dan Sminchisescu, C. (2014). Human3.6m: Large scale

datasets and predictive methods for 3d human sensing in natural environments.IEEE Transactions on Pattern Analysis and Machine Intelligence,36(7):1325–1339.

[3] Kingma, D. P. dan Ba, J. (2014).Adam: A Method for StochasticOptimization.arXiv e-prints, page arXiv:1412.6980.

[4] Martinez, J., Hossain, R., Romero, J., dan Little, J. J. (2017). A simple yeteffective baseline for 3d human pose estimation. In ICCV.