Automated Identification of Gait Abnormalities

Adam Gotlin
Stanford University
SUNetID: agotlin
agotlin@stanford.edu

Apurva Pancholi
Stanford University
SUNetID: apurva03
apurva03@stanford.edu

Umang Agarwal
Stanford University
SUNetID: uagarwal
uagarwal@stanford.edu

Abstract

The goal of this study is to develop a cheap and automated way to assess gait abnormalities to help diagnose and track the progression of neuromuscular disease. Today, physicians analyze walking gait using marker-based motion capture in a motion analysis laboratory or hospital, which is expensive and time consuming for the patient. Patients are scored using the Gait Deviation Index (GDI), which indicates the extent of gait pathology on a scale from 0 to 100. We propose a machine learning model that uses patient video captured by a commodity device (such as a mobile camera phone) to predict GDI. We leverage DensePose, an RCNN that identifies humans in images and refeaturizes the image into a spatial representation of the human's pose. The DensePose output is fed into a machine learning model to predict GDI. The highest performing model was a deep neural network which leveraged a CNN and an LSTM network. The final model predicted GDI with a root mean squared error of 4.2, which is comparable to the state-of-the-art.

Introduction and Background

Several prominent pathologies, such as Cerebral Palsy, Parkinson's disease, and Alzheimer's disease, can manifest themselves in an abnormal walking gait. Abnormal gait is characterized by irregular patterns in step length, step cadence, joint angles, and poor balance. Early detection of these abnormalities can help in diagnosis, treatment, and effective post-treatment monitoring of patients.

A comprehensive metric used to assess the extent of gait pathology is the Gait Deviation Index score, or GDI [1]. By analyzing gait kinematics (the position, orientation, and velocity of body segments over time), physicians score patients on GDI, which is then used to influence decisions on treatment. GDI is a pivotal metric used by biomechanists and physicians for gait analysis throughout the world.

The current gold-standard method of measuring GDI, marker-based motion capture, places a set of reflective markers on body segments and tracks their trajectories over time. Sessions with patients can last multiple hours and cost hundreds or thousands of dollars. A potential less expensive and less time-consuming alternative is to analyze video captured by commodity devices (i.e. mobile camera phone) using machine learning algorithms to predict GDI.

Previous attempts have been made to predict GDI from monocular video footage using a projection of joint centers onto the two-dimensional plane of the camera. In this project, we leverage cutting-edge computer vision tactics to extract three-dimensional features from each frame of video. By stacking processed frames into a video sequence, relevant spatiotemporal features can be modeled for gait characterization. Thus, the project goal is to use monocular video footage to predict GDI score with lower root mean squared error (RMSE) than existing methods.

Related Work

Our work builds on the efforts of many machine learning scientists who developed models to extract spatiotemporal features from video, as well as biomechanists who have analyzed human motion to help physicians assess neuromuscular pathology. A leader in this field is Lukasz Kidzinski, a member of Stanford's Mobilize Center, who was our advisor for this project. We met with Lukasz throughout the project for guidance on data processing and model generation. In 2017, Lukasz and his team used a temporal convolutional network built on videos processed through OpenPose to predict GDI [2]. OpenPose is a system that detects key points of the human body on an image and projects them onto the 2D frame of the camera [3][4][5]. Lukasz's team predicted GDI with great accuracy; his work is considered state-of-the-art (exact performance will not be disclosed as results have yet to be published).

A critical component to our analysis is the refeaturization of images into a spatial representation of human pose. Specifically, we leveraged the DensePose algorithm which converts Red-Green-Blue images to Index-UV coordinates that act as inputs to our model. This algorithm was developed by Facebook AI Research group in February of 2018 [6]. The DensePose RCNN is a leading open-source pose estimation model that assigns each pixel in an image to one of thousands of surface locations on a modeled human mesh. DensePose builds on prior work in human pose estimation, most notably the Skinned Multi-Person Linear Model [7].

Our models and experiments were motivated by researchers who have used machine learning to extract spatial and spatiotemporal features from video. IBM used a CNN with a multi-layer perceptron to classify images into one of many types [8]. Mahendran et al. leveraged a CNN to predict the distance, position, and orientation of an automobile within a continuous regression framework [9]. These efforts helped motivate the spatial component of our model, which was used to extract lower level features for each frame. For our task though, we found processing a single frame via pose estimation frameworks was insufficient to predict GDI with high accuracy. As such, our most successful models incorporated temporal components.

A guiding work for extracting spatiotemporal features from images was Harvey's blog post [10]. Harvey outlines five methods for identifying spatiotemporal features in a video. Our experiments focused on two methods from Harvey's blog: featurizing each frame with a CNN and passing to an RNN and featurizing each frame with a CNN and passing to a 1D CNN along the time dimension. Tran et al. leveraged 3D convolution blocks to extract spatiotemporal features for a variety of video analysis tasks. In his work, he used 3D convolutional networks on 10-frame clips to study pole vault performance, makeup application, and activity recognition [11]. Tran and Harvey's work helped guide our team's decision around kernel size and stride length when convolving over the time dimension. Another guiding work here was Lukasz's team, who leveraged a 1D CNN to predict GDI [2]. While naturally the most relevant based on application, Lukasz's team used a different input feature space (i.e. OpenPose outputs) than our team (i.e. DensePose outputs). Further, to our knowledge, there has been no previous efforts to predict GDI score using DensePose processed images. By combining prior research and our own experimentation, we found that a CNN with LSTM was the most successful framework.

Dataset

Our dataset comprises of ~3,000 videos of patients walking in a room at Gillette Children's Specialty Healthcare Center for Gait and Motion Analysis [12]. This dataset, collected as part of the patient's routine care, is split into 80% training and 20% validation set. The videos have a ground truth GDI (i.e. they are labeled with a GDI score based on assessments from a physician).

The videos have a resolution of 640x480 and are 25 frames per second. Each frame is processed using DensePose, which maps all pixels of an RGB image to the surface of a modeled human mesh [6]. The DensePose-RCNN finds dense correspondence by partitioning the human body surface, assigning each pixel to a body partition and determining the pixel location in 2D parameterization (UV coordinates) of the body part. The parametric surface model that DensePose fits is the Skinned Multi-Person Linear (SMPL) model [7]. A sample image and the regressed correspondence by DensePose-RCNN is shown in Figure 1.

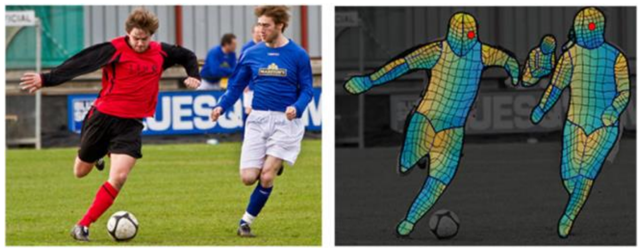

Figure 1: A sample RGB image (left) processed by DensePose (right). Each pixel is assigned to a corresponding point on a human skin mesh [6].

The processed frames consist of three channels - I (Part Index), U and V (Coordinates) - and are passed as inputs to the different learning models. In the case of a model with a temporal component, 10 outputs (i.e. processed frames) are concatenated in sequence per training example.

Before running any initial experiments, substantial work was performed to process data. This included running DensePose algorithm on top of thousands of videos and organizing them into folders. Subsequently, the folders were assigned a GDI score based on the corresponding examID from a joined file of physician assessments. Due to the massive data volume and limits of memory, we did not leverage the entire dataset in our experiments. We typically accessed 500-1,500 videos depending on the model's computational demands. As such, though we considered using video slicing, image mirroring, or other data augmentation techniques, we decided not to implement these as generating additional augmented data was not necessary.

Methods and Experiments

Patient videos are transformed into I-U-V channels by DensePose and are then passed as inputs to our model, which we call GDI-Net. GDI-Net is a machine learning algorithm with spatial and temporal components to predict a patient's GDI score. GDI-Net was implemented in Keras, an open-source neural network library which runs on top of TensorFlow environment [13][14]. The performance of predictions is judged on the basis of minimizing RMSE as compared to the ground truth. The flow of information is presented in Figure 2.

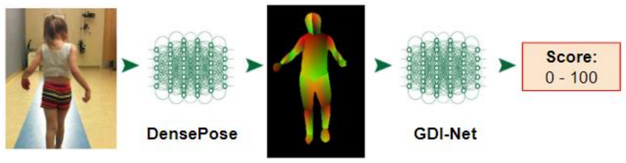

Figure 2: Flow of information for GDI prediction. Video is captured by Gillette Children's hospital and processed by DensePose [15]. These DensePose frames are then passed through GDI-Net to predict GDI score.

Our initial implementations were quick and simple. We built a linear regression model, implemented in Scikit Learn package, using 5 frames of 480x640x3 resolution for each training example [16]. The resulting input vector had a length of 4,608,000, with roughly 435 training examples used to train the regression model and 109 examples used to validate RMSE.

As the model complexity was gradually increased, a sole spatial component was added to GDI-Net architecture. We trained the spatial component, which consisted of VGG16 (an off-the-shelf CNN architecture) or a custom 2D CNN, with 3,834 training frames of 480x640x3 dimension. The model was validated against 951 examples to quantify performance on unseen examples.

The VGG16 model is pre-trained, and only the last layer of the model was replaced by a linear function and trained to perform our regression task. The motivation behind using the pre-trained weights was to investigate the possibility of transfer learning. Since collecting patient data and building custom models is expensive and cumbersome, transfer learning is desirable and could reduce the lead time of any application development. After receiving advice from CS229 course assistants at the course poster session, we also ran a VGG16 model in which all weights were re-trained [17].

A challenge in executing VGG16 was that it requires an image with a standard size of 224x224x3 as an input. Since the DensePose output has a resolution of 480x640x3, the outputs had to be cropped before being passed to VGG16. The crop was made in a manner to preserve as much information as possible (i.e. by selecting pixel values where patients are most likely to appear in the frame); however, some loss is inevitable.

DensePose outputs were passed whole to the custom CNN. The inspiration behind the CNN's architecture was Mahendran et al.'s model that leveraged a CNN to predict the pose of a vehicle within a continuous regression framework. In addition, Lukasz Kidzinski provided instrumental guidance for making important architecture choices in the CNN.

Although the spatial component identifies human poses in a frame, it cannot track the pose trajectories over time. To detect temporal characteristics, a temporal model (either an LSTM or 1D CNN) was added to GDI-Net. The input to a temporal model had 2,920 training examples of 10 frames of 480x640x3 resolution concatenated into a single array. Hyperparameter tuning was performed to further optimize models, mainly tuning learning rate, batch size, and dropout. The majority of this effort focused on tuning hyperparameters for our most promising model, the CNN with LSTM. The complete architecture of the highest performing model is outlined in the Results and Discussion section of this paper.

The immense size of the dataset and the desire to concatenate frames sometimes led to issues with memory overload. In those scenarios, hyperparameters such as batch size and kernel size, were adjusted to avoid memory limits. Further, for computationally expensive models, we leveraged Sherlock, a high-performance computing cluster available to Stanford University affiliates, to reduce run time [18]. More efficient memory management and resource utilization will allow for further data processing and augmentation.

Results and Discussion

A summary of experiments and results is shown in Table 1 below.

Model	**Input Type**	**Training Error**	**Validation Error**
Guess Mean	Frame	13.7	13.7
Linear Regression	Frame	0.0	13.0
VGG16 (pre-trained)	Frame	10.1	9.5
VGG16 (trained)	Frame	11.6	11.4
CNN	Frame	8.1	8.2
CNN + LSTM	**Video**	**1.4**	**4.4**
CNN + 1D-CNN	Video	4.2	11.3

Table 1: Key results from experiments. Models are distinguished by input type, whereby each training examples were either defined as individual frames or by video (i.e. sequence of frames). The error metric reported is RMSE of GDI score.

The initial models, guessing the mean and linear regression, as expected, performed poorly on the validation set. The most successful frame-specific model was a custom-built CNN that was able to reach 8.2 RMSE. The highest performing model overall was a video model that consisted of a CNN and an LSTM network; the model reached RMSE of 4.4 on the validation data.

One of the interesting findings from our experiments was the relatively poor performance from using an off-the shelf model. The initial goal in these experiments was to exploit transfer learning to build complex networks with millions of parameters that are pre-trained. We used VGG16, a model that is readily available within the Keras API and has reported success with image classification tasks [8]. Though this architecture is built for classification, we thought it could still be valuable after replacing the final softmax layer with an appropriate linear activation. The model performed poorly, reaching an RMSE of 9.6. We hypothesize the poor performance was due to a combination of factors: the architecture and parameter weights were tuned with classification in mind; the network expects to receive RGB data by pixel, not IUV coordinates; images were cropped and resized to fit into the VGG16 architecture. To test our first hypothesis, we decided to train the model parameters from scratch. The trained model performed worse than the fixed-weights model. It is possible that better hyperparameter tuning can lead to enhanced performance here. Nevertheless, our current hypothesis is that the cropping and resizing images led to lost information which reduced the ability to predict GDI. Our cropping procedure was to manually scan images and pick a subsection of the 480x640x3 images that we could fit into the 224x224x3 VGG16 input. This had damaging effects to the model. A next set of experiments could involve adding dimension-reducing convolution layers to the beginning of VGG16 or picking a new pre-trained network that better fits our application.

Frame specific models that only captured spatial features of a given frame did not perform well. This is expected, as GDI is largely determined by trajectories of body parts, and individual frames do not hold temporal information describing how the patient moves over time. As such, we spent most of our time and effort experimenting with spatiotemporal models.

The best performing model combined a 2D CNN on each frame and an LSTM to capture temporal patterns. The learning curve, regression plot, and detailed architecture are outlined in Figure 3. We believe this model had the greatest performance because GDI is based on trajectories, and incorporating an LSTM allowed the model to identify time-based patterns which ultimately translate into trajectories. We also leveraged much of the CNN architecture of our best-performing frame model, which we believe did a good job featurizing each frame into a relevant vector to be sequenced into the LSTM. The final model used an initial learning rate of 0.01 which decayed linearly to 0.001 at the end of 100 epochs. As seen in the learning curve, the model is overfit to the training data. Further hyperparameter tuning, such as

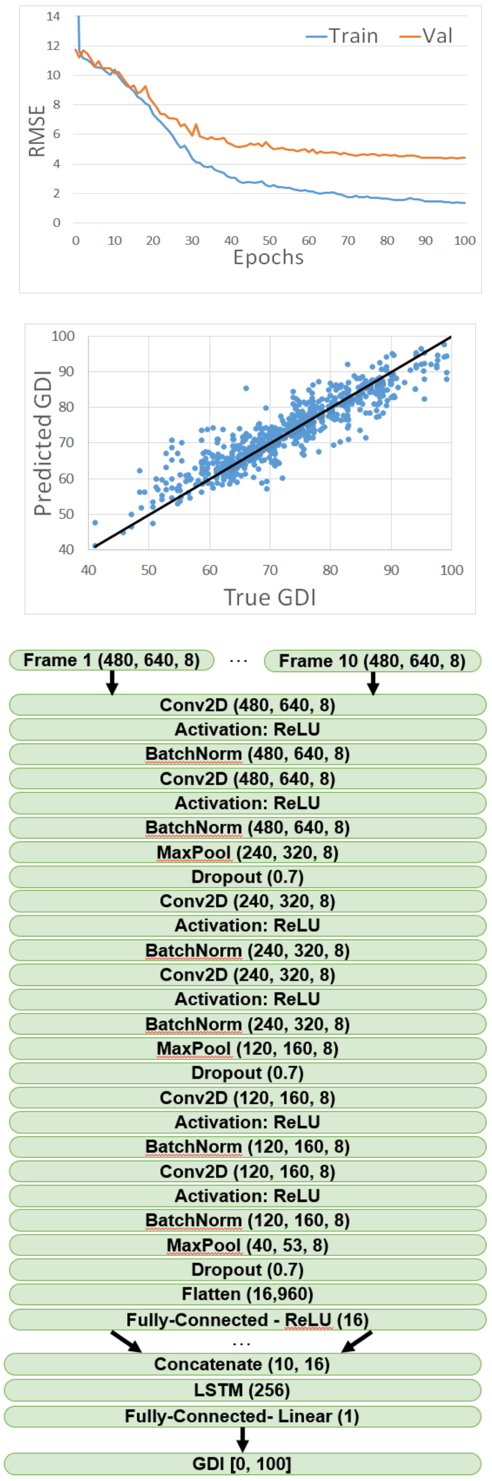

Figure 3: (Top) Learning curve for top performing model. (Middle) Regression plot comparing True GDI vs. Predicted GDI on the validation set. (Bottom) Architecture of top performing model. Each frame is separately processed by the 2D CNN and concatenated in sequence before passing to the LSTM.

increasing dropout, could help reduce this overfit. Another shortcoming worth highlighting is that we did not train the model on the full dataset due to limited memory. We also used a small batch size of 4 to allow our machines to process each calculation, which resulted in a more stochastic model than desired. If we implemented better techniques to store and manipulate large matrices, we could train on a larger dataset and potentially reduce overfit.

Nevertheless, the results achieved by the best model are promising. As seen in the regression plot comparing true GDI to predicted GDI, the model successfully captured the variance in GDI scores, with an R^2 of 0.86. Lukasz Kidzinski and his team, who spent months tuning a model based on OpenPose pre-processing, was able to achieve a slightly lower RMSE on a similar dataset (exact performance will not be disclosed as results have yet to be published). Though our models did not outperform the cutting edge, our results suggest DensePose-based algorithms have the potential to compete with state-of-the-art techniques.

Conclusions and Future Work

In this study, we implemented a machine learning approach to develop a cheap and automated way to assess gait abnormalities. The results, especially that of a CNN with LSTM model, are very promising. We are excited by our model's potential to help diagnose and track the progression of neuromuscular disease.

Although the CNN with LSTM model was able to achieve a low RMSE, error analysis should be performed to investigate the deviations between the model and ground truth. Once the nature of these errors is understood, the model and the architecture can be fine-tuned for better accuracy.

An option to enhance our architecture is to implement 3D convolution blocks instead of a separate spatial and temporal component. We hypothesize that it may perform better than the spatiotemporal model, as it can capture lower level features in time and is not affected by the way data is passed from the spatial component to the temporal component.

In our experiments, we observed a growing gap between training error and validation error as training progressed, which suggests overfitting. Although we attempted to mitigate this using dropout, more could be done to generalize our model. Future experiments can focus on reducing model complexity or incorporating L2-regularization.

An interesting approach we would like to implement is building a classification network by bucketing GDI scores to the nearest integer. Using a softmax layer, we can take the probability-weighted sum of bucket values to determine a scalar GDI score. In other words, we can train the network as a classification task but derive the scalar GDI score using the appropriate weighted sum of the softmax output. This option also opens the opportunity to use off-the-shelf classification frameworks for our task.

Our capacity to experiment was constrained by memory overload issues. The efficient management of memory and resource utilization would allow for more rapid experimentation. Chunking, memory swapping, or simply accessing machines with larger random-access memory can be applied to address this issue.

Future experiments should explore refeaturizing the processed DensePose outputs to global X-Y-Z coordinates. This would allow us to manually engineer additional relevant features, such as knee flexion angle, that are expected correlates of GDI score. We can further compress our data by considering only the X-Y-Z coordinates of the most relevant body landmarks, as movements of the hip, knee, and ankle are particularly important for gait analysis. This process would require the manipulation of DensePose outputs to a customized SMPL human body model.

The potential for future work is enormous as we have just scratched the surface of DensePose's capabilities. A determined effort can lead to the development of a robust, reliable, and low-cost alternative to analyzing human gait.

Contributions

Adam Gotlin collaborated closely with the Stanford Mobilize Center and other project partners to secure data and share knowledge. He orchestrated meetings with advisors and experts and helped identify best practices for time-series and movement data. He led efforts on building temporal CNN models.

Apurva Pancholi spearheaded initial experiments and was involved in data preprocessing. He owned data transfer from the Mobilize Center to our project team. Apurva developed a custom CNN that was the primary spatial component for our highest performing model.

Umang Agarwal led data processing and consolidation. He owned efforts to read and interpret DensePose source code in order to generate GDI-Net model inputs. He led efforts to exploit transfer learning and contributed to tuning neural network models to maximize performance.

Code

Code for this project can be found at: https://github.com/agotlin/CS229DP

References

[1] Michael H. Schwartz and Adam Rozumalski (2008). "The gait deviation index: a new comprehensive index of gait pathology". *Gait & posture*, 28(3):351-7, 2008.

[2] Łukasz Kidziński, Bryan Yang, Jennifer Hicks, Scott Delp, Michael Schwartz (2018). "Automatic diagnostics of gait pathologies using a mobile phone". technical report, Stanford 2018.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh (2017). "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields". *CVPR*, 2017

[4] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh (2017). "Hand keypoint detection in single images using multiview bootstrapping". *CVPR*, 2017.

[5] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh (2016). "Convolutional pose machines". *CVPR*, 2016.

[6] Riza Alp Guler, Natalia Neverova, Iasonas Kokkinos (2018). "DensePose: Dense Human Pose Estimation In The Wild". *CVPR,* 2018.

[7] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, Michael J. Black (2015). "SMPL: A Skinned Multi-Person Linear Model". Max Planck Institute for Intelligent Systems.

[8] Apache Software License, Version 2 (2018). "Image Classification Using CNN and Keras". GitHub repository, https://github.com/IBM/image-classification-using-cnn-and-keras

[9] Siddharth Mahendran, Haider Ali, Rene Vidal (2017). "3D Pose Regression using Convolutional Neural Networks". Center for Imaging Science, Johns Hopkins University.

[10] Matt Harvey (2017). "Five video classification methods implemented in Keras and TensorFlow". *Coastline Automation,* 2017.

[11] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri (2015). "Learning Spatiotemporal Features with 3D Convolutional Networks" *CV*, 2015.

[12] The authors acknowledge the staff of the Center for Gait and Motion Analysis at Gillette Children's Specialty Healthcare for collection and curation of the subject data.

[13] Martín Abadi et al. (2015). "TensorFlow: Large-scale machine learning on heterogeneous systems". Software available from tensorflow.org.

[14] Francois Chollet et al. (2015). "Keras". Software available from keras.io.

[15] Hanson, Nick (2017). "Kids Health Matters." *Gillette Children's Specialty Care,* 2017, www.gillettechildrens.org/khm/topics/gait-analysis.

[16] Fabian Pedregosa et al. (2011). "Scikit-learn: Machine Learning in {P}ython". *Journal of Machine Learning Research,* 12:2825-2830, 2011.

[17] The authors acknowledge the CS229 course staff who provided advice and guidance throughout this project.

[18] Some of the computing for this project was performed on the Sherlock cluster. We would like to thank Stanford University and the Stanford Research Computing Center for providing computational resources and support that contributed to these research results.

[19] Angjoo Kanazawa (2018). "Perceiving 3D Humans in the Wild". Stanford, CA.

[20] Jason Brownlee (2018). "Get the Most out of LSTMs on Your Sequence Prediction Problem". Available from https://machinelearningmastery.com/machine-learning-with-python/.

[21] Buitinck et al., (2013). "API design for machine learning software: experiences from the scikit-learn project".