CS230

FallDetectNet: A Computer Vision Platform for Fall Detection

Yanichka Ariunbold
Department of Computer Science
Stanford University
yanichka@stanford.edu

Stephanie Brito
Department of Computer Science
Stanford University
sbrito@stanford.edu

Ariel Leong
Department of Computer Science
Stanford University
akl1@stanford.edu

Abstract

Falls are the leading cause of fatal and nonfatal injuries among the elderly. [1] In order to help alleviate this health crisis, we introduce FallDetectNet, a computer vision model that detects falls in real time using images obtained from depth sensors. Our approach incorporates convolutional neural networks, data augmentation, and transfer learning in order to classify whether depth sensor images represent a fall or not. Using transfer learning, we were able to achieve an accuracy of 99% and a recall score of 99% on the test set.

1 Introduction

According to the U.S. Centers for Disease Control, over 2.8 million adults over the age of 65 are injured in a fall annually, leading to approximately 800,000 hospitalizations and 27,000 deaths. [1] Falls exponentially increase with age-related biological changes, leading to a high occurrence of fall-related injuries in aging societies. The number of injuries caused by falls is projected to increase by 100% in 2030. [2] Thus, early detection of falls in a manner that respects people's privacy would be helpful in mitigating falls' negative health consequences. In this project, we input depth images into a convolutional neural network to output a prediction of whether a fall occurred in the image.

2 Related Work

The fall detection task is normally done by hand; namely by passersby who happen to see an injured elderly person. Research has been done on more automatic and continuous ways of detect falls. One set of approaches involves inputting hand-crafted features into machine learning classifiers. One approach used features from RBG (color) images such as the ratio of the height and width of the bounding box drawn around a person. A k-nearest neighbors classifier was used and achieved an accuracy of 84.44%. [3] Similarly, another group used accelerometer and gyroscope data to calculate features such as the range of angular velocity and a support vector machine classifier that achieved a specificity of 99.5% and sensitivity of 97%. [4] Although the ingenuity involved in devising hand-crafted features is admirable and these models perform well compared to others when data is scarce, it is difficult for hand-crafted features to generalize to all types of falls. [4] The above methods

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

are different algorithmically but share our concern for privacy. Part of the reason hand-crafted features were used was to use the data in a way that removed identifiable features. [3]

Another category of approaches, convolutional neural networks (CNNs), are state-of-the-art because they do not require expensively hand-crafting features and achieve high accuracies. But, their performance is limited by the amount of data available. Some CNN models share similarities but use different data sources than ours. One group also used transfer learning. They input accelerometric data into an AlexNet architecture that was pretrained on the ImageNet dataset. It achieved an accuracy of 96.43% and precision of 95.83%. [5] Another group modified an AlexNet and took RBG images as input, achieving an accuracy of 99.98%. [6] Although these methods achieved high accuracies, the data they rely on might not be as usable as depth data. The accelerometers and gyroscopes have to be worn by a person, which can be inconvenient. It is difficult to preserve patient privacy in the RBG images. An additional group used depth data like we did, though they used videos and thus a 3D-CNN. Though videos can ostensibly be more accurate in gauging whether a fall occurred, as it relies on multiple frames rather than a single one and usually uses more complex models [7], this may make it slower to detect a fall. Also, fewer fall detection videos are available than images. This group overcame this obstacle by using data augmentation, increasing the model accuracy from 69.6% to 92.4%. [8] In this work, we hope to use a source of data that was not used in the literature, individual depth images, but that seems to have privacy and speed advantages. We will use a combination of the techniques mentioned here: CNNs, transfer learning, and data augmentation.

3 Dataset

The primary data we used are depth images from the UR Fall Detection Dataset (Figure 1). [9] Depth images contain information about the distance of objects from a camera. They help to preserve privacy since it is difficult to identify people from them. We used 4013 depth images from the UR Fall Detection Dataset for training and 1977 for testing. They contains images obtained from two Microsoft Kinect cameras at different angles for 30 distinct falls. Each fall has about 150 labeled frames. The image format is PNG16; the image has dimensions 480x640x3. Each frame is labeled as follows: -1 for not a fall, 1 for a fall and 0 for a temporary pose (the person is about to fall).

(a) Fall (b) Not a fall

Figure 1: Examples of data from the UR Fall Detection Dataset

3.1 Data pre-processing

To pre-process the data, we first downloaded all of the fall and not-fall images. We then applied min-max normalization and resized the images to the shape (224, 224, 3), which is required to use the VGG-16 model with top layers removed.

4 Methods

4.1 Convolution Neural Network

We trained a simple CNN on all of our data. We chose a CNN because these networks are commonly used for image classification tasks as the convolutional layers extract feature maps from images that can help with the final prediction task. We used two convolutional layers of size 64, then 32 with filter sizes of 3 and a ReLU activation. We ended the model with a flatten layer and a dense fully connected layer with a sigmoid activation.

4.2 Data Augmentation

In order to increase the diversity in images of falls and not-falls our model sees, we implemented data augmentation in our model using the "ImageDataGenerator" class in the Keras library to create modified versions of our images. In each epoch of training, the generator returned unique transformed versions of the original images in the dataset. After experimentation with multiple different rotation arguments (10, 15, 30, 45, 60, 90), we decided to use a rotation argument of 15 for higher accuracy. Each image was thus randomly rotated clockwise by a given number of degrees from 0 to 15. Through a similar process of testing, we also controlled the range of horizontal and vertical shifts possible with width shift and height shift arguments of 0.1. Refer to Appendix A for examples of transformed images via data augmentation.

4.3 Transfer Learning

Since our CNN was overfitting, we decided to use transfer learning to help our model generalize beyond our small training set and perform better on the test set. In transfer learning, layers of a model trained on one task are used in a model intended for a similar task. The hope is that some of the knowledge the old model used to complete its old task (as encoded in model weights) will be useful on the new task. We pretrained a VGG-16 architecture on the ImageNet dataset to learn a feature extractor. [10, 11] We chose this dataset and model because it generalizes well to other tasks and has fewer parameters compared to other models, which can lead to faster training with less memory usage. [12, 13] We replaced the VGG-16's fully-connected layers with an average pooling layer and dense layer with sigmoid activation. The remaining part of VGG-16 is comprised of five "blocks" of convolutional layers followed by a pooling layer. We experimented with freezing different numbers of blocks to improve model performance.

4.4 Hyperparameters

All of our models used the same hyperparameters, including the learning rate, number of filters in convolutional layers, and number of training epochs. We used choices that performed well on related work [12, 6] and chose to focus on varying the number of frozen layers in our transfer learning model. Our loss function was binary cross entropy loss as shown in Equation 1.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{1}$$

5 Results + Discussion

5.1 Evaluation Methods

To evaluate our model, we looked at accuracy, precision, and recall. As with most tasks related to healthcare, the consequences of false negatives are much higher than those of false positives. Thus we place particular emphasis on recall.

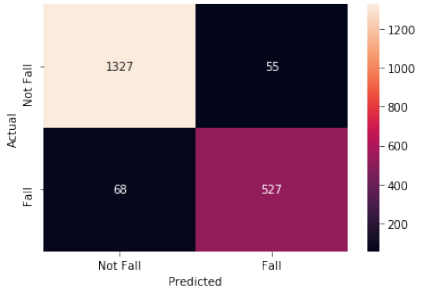
5.2 Convolution Neural Network

The baseline CNN model performed well with a training accuracy of 98.55% and a test accuracy of 93.77%. As shown in Figure 2b, the model had more false positives than false negatives, which is good for our desired task. When we applied data augmentation to the model, it actually performed worse. This is an indication that while our results on our initial dataset were good, our model is probably not generalizable. See the appendix for examples of how the model performs on new images.

Figure 3 shows the results of the CNN with data augmentation. The accuracy on the training set was 98% and the accuracy on the test set was 73%. As Figure 3 shows, this model has the largest difficulty distinguishing between actual falls and not falls, resulting in an incredibly low recall score of 0.44. In short, we have many false negatives, but not too many false positives, which is not suitable for this task. This is probably because there are more images that aren't falls in the dataset. Therefore,

	Data aug	No data aug
Train acc	.7301	.9855
Test acc	.7511	.9377
Precision	.71	.91
Recall	.44	.89
F1	.55	.90

(a) Metrics



(b) VGG-16 architecture

Figure 2: Evaluation of CNN with and without data augmentation

the model doesn't make strong predictions in either direction and will default to not-fall more often to attain a lower loss.

5.3 Transfer learning

Our reasons for choosing the transfer learning model we did are explained in the "Methods" section. After seeing the poor performance of the transfer learning model when all of the layers were unfrozen, we varied the number of blocks frozen. As shown in Figure 3, the best performance was achieved when the first three or first four convolutional "blocks" were frozen. This intermediate number of blocks seems to achieve a good balance: keeping the earlier layers frozen will maintain the weights that are good at capturing more generic image features, which would assist with our image classification problem. But if too many layers are frozen, then the model cannot adapt to our specific fall detection dataset. The transfer learning model performed better than the original CNN, both with and without data augmentation. Transfer learning may have helped the model generalize and thus perform better on the test set (this is also supported by the smaller gap between training and test dataset accuracies for the transfer learning model compared to the original CNN). As shown in Figure 4b, the transfer learning model tended to misclassify not-falls as falls more than the reverse. This could be due to the reason mentioned in the above section, namely that there are more not-fall images in the dataset.

Data augmentation did not improve model performance; the train accuracy, test accuracy, precision, recall, and F1 score were 0.9905, 0.9863, 0.98, 0.98, and 0.98, respectively. It could be that the augmentations were not needed in this particular case. With respect to the training images, the test dataset images were not rotated and were not vertically or horizontally shifted since they were taken using the same camera in the same room.

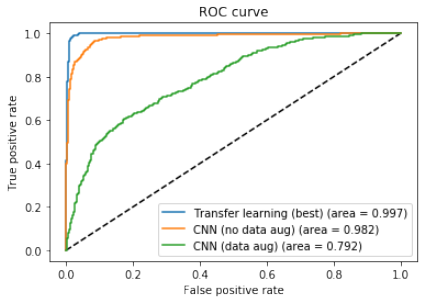
Layers frozen	Train acc	Test acc	Precision	Recall	F1
None	.6935	.7086	0.00	0.00	0.00
1-3 (block 1 frozen)	.6935	.7086	0.00	0.00	0.00
1-6 (blocks 1 and 2 frozen)	.8139	.8260	0.86	0.48	0.62
1-10 (blocks 1,2,3 frozen)	.9599	.9737	0.93	0.98	0.96
1-14 (blocks 1,2,3,4 frozen)	**.9858**	**.9863**	**.94**	**0.99**	**0.97**
1-18 (all blocks frozen)	.7907	.8154	0.82	0.51	0.63

Figure 3: Evaluation of transfer learning model with top results bolded

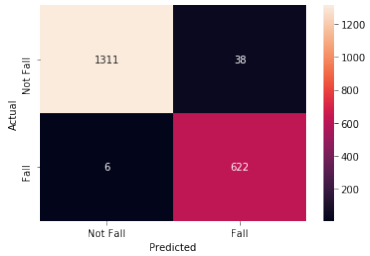
5.4 Error Analysis

As we can see in the receiver operating characteristic (ROC) curve in figure 4a, the transfer learning model performed best with an area under the curve of 0.997. The CNN with data augmentation performed worst with an area under the curve of 0.792. This plot shows that the transfer learning approach was far superior in this case to the baseline CNN in identifying true positives.

Figure 5a shows an image that was misclassifed as a non-fall by the CNN with data augmentation. We believe that the model is not robust enough to properly distinguish between classes in the presence



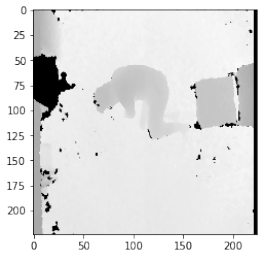
(a) ROC Curve for selected models



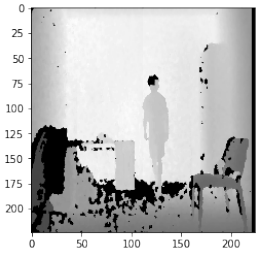
(b) Confusion matrix for best transfer learning model

Figure 4: ROC curve for selected models and confusion matrix for best transfer learning model

of furniture. Figure 5b shows a false positive. Depth sensors naturally experience a lot of noise, especially when there is a lot of movement and changes in lighting. In this example, we believe the model predicted it was a fall because of the noise in the image coming from the black dots on the floor, even though the person is clearly standing.



(a) False negative from CNN with data augmentation



(b) False positive from CNN with no data augmentation

Figure 5: Examples of prediction errors

6 Conclusion + Future Work

We achieved a model with similar performance to the "Related Work" section's with a more practical source of data. Our transfer learning model, either with or without data augmentation, performed the best. The baseline CNN model without data augmentation did not perform as well, though it still achieved high evaluation metrics. The baseline CNN model without data augmentation performed the worst. We think that the transfer learning model performed well because pretraining on another dataset helped it generalize beyond the small training dataset. The baseline model did not perform as well since it was only trained on the small UR Fall Detection dataset. A possible reason why the baseline CNN with data augmentation performed the worst is that the depth images were already noisy, and the modifications to the images could have made it even harder for the model to distinguish between fall and not-fall images. If we had more time, we would address the large disparity between precision and recall values that some of our models displayed. Since the disparity implies that the model has a higher number of false negatives than false positives, we would address the problem by creating a custom loss function that would strongly penalize false negatives. Furthermore, as the results in the appendix and from data augmentation show, our model does not perform well when given images outside of our initial dataset. To combat this, we would find new datasets for the same task to create a more robust model.

7 Contributions

All group members contributed equally to the project. Ariel focused on transfer learning, Stephanie focused on the CNN and Yanichka focused on the data augmentation.

8 Code

Our code is available in our public Github repo linked here.

9 Appendix A

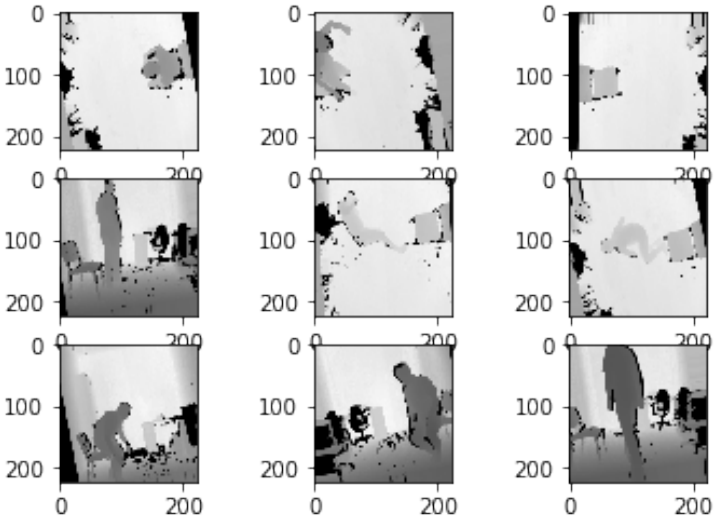
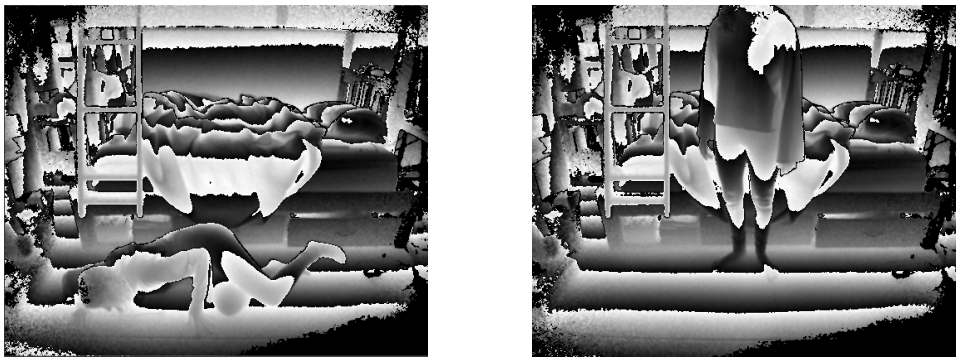


Figure 6: Examples of data augmentation transformations

10 Appendix B

To experiment with the generalizability of our models, we created our own data using an Xbox Kinect sensor and Microsoft SDKs. Our model was unable to properly categorize these images, which showed that our model is not as robust as it needs to be to fully function in the real world. These samples also show how noisy depth sensor images can be in rooms with a lot of furniture/lighting.



(a) Fall image generated by our team

(b) Non fall image generated by our team

Figure 7: Examples of prediction errors

References

[1] Gwen Bergen, Mark Stevens, and Elizabeth Burns. Falls and fall injuries among adults aged ?65 years - united states, 2014. *MMWR. Morbidity and mortality weekly report*, 65:993–998, 09 2016.

[2] Plaza I Igual R, Medrano C.

[3] Chien-Liang Liu, Chia-Hoang Lee, and Ping-Min Lin. A fall detection system using k-nearest neighbor classifier. *Expert systems with applications*, 37(10):7174–7181, 2010.

[4] Naohiro Shibuya, Bhargava Teja Nukala, Amanda I Rodriguez, Jerry Tsay, Tam Q Nguyen, Steven Zupancic, and Donald YC Lie. A real-time fall detection system using a wearable gait analysis sensor and a support vector machine (svm) classifier. In *2015 Eighth International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, pages 66–67. IEEE, 2015.

[5] Haben Yhdego, Jiang Li, Steven Morrison, Michel Audette, Christopher Paolini, Mahasweta Sarkar, and Hamid Okhravi. Towards musculoskeletal simulation-aware fall injury mitigation: transfer learning with deep cnn for fall detection. In *2019 Spring Simulation Conference (SpringSim)*, pages 1–12. IEEE, 2019.

[6] Xiaogang Li, Tiantian Pang, Weixiang Liu, and Tianfu Wang. Fall detection for elderly person care using convolutional neural networks. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6. IEEE, 2017.

[7] Hongfei Li, Steven Cryer, Lipi Acharya, and John Raymond. Video and image classification using atomisation spray image patterns and deep learning. *Biosystems Engineering*, 200:13–22, 2020.

[8] Seokhyun Hwang, DaeHan Ahn, Homin Park, and Taejoon Park. Maximizing accuracy of fall detection and alert systems based on 3d convolutional neural network. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 343–344. IEEE, 2017.

[9] B. Kwolek and Michal Kepski. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer methods and programs in biomedicine*, 117 3:489–501, 2014.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] Mohammad Ashraf Russo, Laksono Kurnianggoro, and Kang-Hyun Jo. Classification of sports videos with combination of deep learning models and transfer learning. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–5. IEEE, 2019.

[13] Yuqing Gao and Khalid M Mosalam. Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768, 2018.