Classification of Dance Styles Using CNNs

Henry Lin
Symbolic Systems
Stanford University
henryln1@stanford.edu

Christopher Vo
Computer Science
Stanford University
cvo9@stanford.edu

Abstract

In recent years, deep learning, specifically Convolutional Neural Networks (CNNs), have generated promising results in the field of image classification. We build upon previous CNN architectures and results and extend the work into the area of human motion classification, specifically for different dance styles. For our project, we use the Let's Dance Dataset, a dataset of 1000 videos split evenly into 10 different classes for classification. We explore the problem through the lenses of 3 different deep learning architectures and elucidate the most promising areas for future work. We began with a straightforward 2D CNN using single frames taken from the video and built up to 3D convolutions and a CNN-RNN hybrid model. Ultimately, we reach a peak F_1 score of 0.40 and accuracy of 0.41 on the test set using 3D convolutions.

1. Introduction

The recent advent of deep learning in the field of computer vision has led to an increased interest in the applications of Neural Networks, with breakthroughs in many different areas such as image recognition, style transfer, and image captioning. For example, state-of-the-art models have exceeded human performance on the ImageNet challenge. In our project, we aim to classify different dance forms from images taken from video frames. Dance is an ideal application to study the potential for using deep learning to understand motion due to the unique mix of diversity and similarity between various dances. For instance, different dances such as Waltz and Tango have many elements in common with each other yet also contain distinguishing poses that clearly separate them. We also look at dance styles that exist outside the traditional ballroom studio such as breakdancing and ballet, both of which are popular dances. We believe that classifying dance motion provides an excellent case study into the feasibility of capturing general motion from images and videos.

Specifically, we plan to develop and experiment with two more traditional CNN architectures incorporating both 2D and 3D convolutions and an experimental model combining CNN and LSTM architectures and evaluate performance on the task of classifying dance images from videos. For our traditional 2D CNN architecture, we plan to input static images and output the predicted dance. For our 3D CNN and CNN-LSTM models, we plan to input series of static images to represent frames in a video and output the predicted dance. This experiments with the feasibility of capturing motion from videos and not just static images. Much of our inspiration and dataset comes from a paper by Castro et al. [4] which attempts to classify dance images from videos by experimenting with various neural network architectures. We hope to build on this work by trying these different model architectures and feature representations and evaluating the effect of these changes on performance.

2. Related Work

Many papers have introduced various approaches and architectures for image and video classification, as well as models that combine CNN and LSTM models to capture temporal information. These papers range from general labeling of images and videos to domain specific tasks such as emotion detection [6] or speech modeling [10].

For image classification, many state-of-the-art approaches have managed to perform very well, some even exceeding human performance on the ImageNet challenge. Our more traditional 2D CNN model draws inspiration from two of the more simplistic yet powerful of these popular architectures: AlexNet [9] and VGGNet [11]. These architectures are appealing due to their relatively lower number of layers compared to newer approaches, their simple sequential layer structure, and the use of smaller filters and pooling size in the case of VGGNet.

Many approaches also attempt to extend existing work on static image classification to classify and label videos. The biggest challenge of video classification is finding

approaches to represent the temporal dimension and combine the various frames of a video. In a paper by Karpathy et al. on video classification [8], various temporal fusion models were introduced that have become very influential in the area of video processing. Our 3D CNN architecture incorporates the idea of early fusion presented. Ng et al. [14] attempt to extend this work by also experimenting with temporal pooling and explicitly representing videos as sequences of frames.

Additionally, some papers attempt to capture temporal information by combining CNN and LSTM architectures for various tasks. While convolutional networks are able to capture spatial information, recurrent neural networks (RNNs) are able to capture temporal information. Long short-term memory (LSTM) networks are a special type of RNN that handle the problem of vanishing/exploding gradients that plague traditional RNNs. Donohue et al. [5] combine CNN and LSTM architectures to label videos by passing individual frames through a CNN to obtain a feature representation and then use the LSTM to combine the frames as a sequence. Fan et al. [6] extended this idea by combining this model with 3D CNNs and SVMs to classify emotion in videos. Sainath et al. [10] apply this combined architecture with fully connected layers to various speech recognition tasks. Our experimental CNN-LSTM architecture draws from this base idea of combining CNNs and LSTMs to represent both the spatial and temporal dimensions of videos.

Various CNN approaches have also been applied to capture motion in images and videos. In a paper by Teney and Hebert [13], an approach that combines CNNs with methods to extract optical flow from videos is used to specifically classify motion videos. This approach takes advantage of the fact that motion is a large part of the videos so domain knowledge about optical flow can be used with greater effect that traditional videos. The most relevant paper to our project is one by Castro et al. [4] titled *Let's Dance* that specifically attempts to classify dance videos with CNN architectures. Our dataset of dance videos as well as some model inspiration comes from this paper. In addition, the *Let's Dance* paper gives us an idea about the feasibility of our task of classifying dance forms from images and videos. However, we hope to provide a more thorough evaluation and analysis than [4], which just uses basic accuracy as their only performance metric.

3. Dataset

To train and test our model, we used the *Let's Dance* dataset from [4]. It is a comprehensive dataset of dance videos split into 10 different classes, each class containing 100 videos. The dataset is preprocessed into the individual frames of a video, and with each video being 10 seconds long at 30 frames a second; each video is represented in the dataset as 300 frames. Overall, there are 300,000 total image frames that were split 80-10-10 into training, validation, and test. Depending on the original quality of the video, each frame was either 720 x 1280 or 1080 x 1920. To deal with variable frame sizes, we downsized all images to either 144 x 256 or 216 x 384, depending on the computational constraints and the given model. In additional to this, we performed data augmentation in our final model by performing horizontal flips on the frames.

(a) Image, 1080 by 1920 (b) Image, 216 by 384

Figure 1: Training Image Example, Breakdance

(a) Image, 1080 by 1920 (b) Image, 216 by 384

Figure 2: Validation Image Example, Waltz

In Figure 1 and 2, we present two example frames we used during the training and validation of our model. It is clear that despite the downsizing of the image, the overall objects and individuals in the photo are still relatively clear. Thus, down-sampling images was an important step in reducing the memory and computational burdens while maintaining high performance. We tested this aspect by trying

Figure 3: Multiple Frames from Video, Swing

two different image sizes, as mentioned above, which we will discuss later. In Figure 3, we provide an example of how a video is split up into multiple frames in the dataset. Since a single second of a video already has 30 frames, we skipped frames in an attempt to capture more meaningful changes between selected frames. Figure 3 shows the 1st, 11th, 21st, and 31st frames from a video in the swing class from the validation set going from top left to bottom right.

4. Methods

4.1. Single Frame Convolutions

Our first architecture draws inspiration from some of the more traditional CNN architectures for image classification, specifically a combination of AlexNet [9] and VGGNet [11]. This CNN-based model takes in single image frames as inputs and classifies which of the ten dance forms the image belongs to, similar to standard image classification tasks such as the ImageNet Challenge.

Specifically, our CNN architecture can be seen in Figure 4 and is as follows. The architecture consists of three sets of 2D convolutional layers and a max-pool layer. The first set of layers consists of two 2D convolutional layers with 64 3×3 filters with a stride of 1. Each of these convolutional layers is followed by a batch normalization layer and all of these layers are followed by a 2×2 max pooling layer with a stride of 2 and then a dropout layer with a keep probability of 0.8. The second set of layers is the same as the first set except that the convolutional layers contain 128 filters each. The third set of layers is also similar but there are three convolutional layers with 256 filters each and three corresponding batch norm layers. The final layer is a fully connected layer which outputs the scores for each of the ten dance classes.

We chose to model our architecture off of AlexNet and VGGNet due to the relatively low number of layers and parameters compared to deeper models such as ResNet, and the simpler sequential structure of these models. Keeping the number of layers low reduces the number of parameters and allows us to train the model more efficiently. From AlexNet, we incorporated the ideas of using ReLU non-linearities as well as normalization layers. ReLU is picked as the non-linearity as it is a commonly used activation function that is computationally inexpensive, does not saturate in the positive region, and works well in practice while batch normalization is a popular method that makes the model more robust to bad initializations. VGGNet inspired our structure of sets of two or three convolutional layers followed by a max pooling layer and also our choices for filter size and pool size. The smaller filter size (3×3) was chosen as multiple smaller filters have

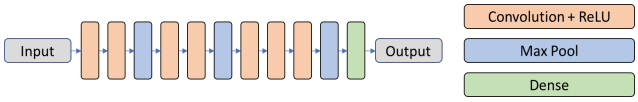


Figure 4: Visualization of CNN Model

the same effect as larger filter sizes with less parameters, allowing training to be more efficient and reducing the risk of overfitting. Also similar to VGGNet, the number of filters used per convolutional layer doubles with each set of layers, which allows the model to capture more detailed feature representations.

The model is trained using Adam optimizer with softmax cross entropy as the loss function where the total loss is the mean of each training image's individual loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}})$$

where there are N training images in the batch and f_{y_i} is the correct class score for the image and f_j is the class score for class j. L_2 regularization is applied to the loss. Batch size of 128 is used and the model was trained for 5 epochs with 1000 iterations each. The model was evaluated on the validation set every epoch.

4.1.1 Batch Normalization [7]

Due to the relatively small individual image examples for the single image classifier, we employed batch normalization as a means to help reduce training time and increase the model's resistance to bad parameter initialization. It does this by enforcing a Gaussian distribution throughout the network. To do this, we added a batch normalization layer after each convolution in Figure 4.

4.2. Multiple Frame Convolutions

This architecture incorporates 3D convolutions to take in sets of image frames as inputs and outputs which of the 10 dance forms the set of images belongs to. The set of images represents a video and linking multiple images frames together is an approach to capture the temporal dimension present in videos.

The specific architecture for our 3D CNN is the same as for our 2D CNN in Figure 4. However, there is one key difference: the convolutional and max-pooling layers are 3D rather than 2D. In our 2D convolutions, the filters have size 3×3 and convolve over the height, width, and color channel dimensions of the static image. In 3D convolutions, since we are classifying sets of images rather than a single image, there is an additional dimension to convolve over, a

dimension of the number of images frames in an image set. Therefore, our 3D convolutional layers contain filters with size $3 \times 3 \times 3$ with stride of 1. Our 3D max-pooling layers have pool size $1 \times 2 \times 2$ with stride 1 for the image set dimension and stride 2 for the width and height dimensions as we want to keep the size of our image set dimension the same. With more parameters, we also lower our dropout keep probability to 0.7 to reduce overfitting. Otherwise, our architecture structure is the same as with the 2D CNN. Using 3D convolutions is similar to the idea of early fusion for video classification proposed by Karpathy et al [8]. Similar to the 2D CNN, Adam optimizer with softmax cross entropy loss is used to train the model, but due to drawing multiple images per example, the batch size was reduced to between 4 and 8. The model was trained for 5 epochs with 1000 iterations each and evaluated on the validation set at every epoch.

4.3. CNN-LSTM Combination

Our third architecture experiments with combining CNN and LSTM architectures to classify dance videos. Similar to our 3D CNN, this model takes in sets of image frames as inputs and outputs the dance form corresponding to the image set. LSTMs have been applied to capture temporal information in many fields such as natural language and video processing so we wanted to evaluate its effect on our task of classifying dance videos. Our architecture was inspired from related works combining these two architecture types such as [5], [6], and [10]. The model takes in a set of images as inputs and feeds each individual frame through a 2D CNN to produce feature maps for these frames. Then, these feature maps are fed sequentially to an LSTM network to produce a classification on the dance of the set of images.

The specific model architecture can be seen in Figure 5 and is as follows. Input image sets are reshaped to concatenate frames together as individual images to feed into a 2D CNN architecture. This CNN architecture is the exact same as our 2D CNN without the final fully connected layer. Instead, the inputs are reshaped again to obtain the original image set representations back. These sequential image set feature representations are then fed into an LSTM layer which outputs the scores for each dance for the image set. Similarly to the other architectures, Adam optimizer is used to train the model with softmax cross entropy as the loss function over 5 epochs of 1000 iterations each. Evaluation on the validation set occurred after each epoch and similar to the 3D CNN, due to inputting sets of multiple images, a batch size between 4 and 8 image sets was used for training.

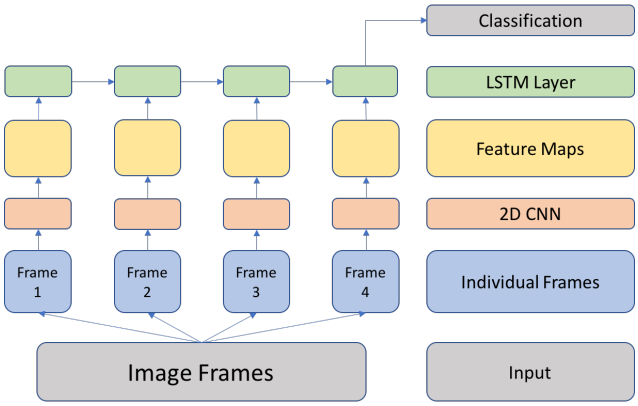


Figure 5: Visualization of CNN-LSTM Model

5. Results and Discussion

5.1. Performance

To evaluate the performance of our models, we used four metrics: precision, recall, F_1 score, and accuracy. Precision and recall are common metrics used in classification tasks that are based on the number of true positives TP (correctly classified instances of the specified class), false positives FP (instances incorrectly classified as that class), and false negatives FN (instances of that class that are incorrectly classified as not being in that class). Specifically, precision is the fraction of instances classified as the class of interest that truly belong to that class and can be defined as:

$$P = \frac{TP}{TP + FP}$$

while recall is the fraction of instances of the class of interest that are correctly classified to belong to that class and can be defined as:

$$R = \frac{TP}{TP + FN}$$

These definitions of precision and recall can be generalizable to multi-class classification given a standard confusion matrix C in which we can easily obtain the precision and recall for a class i [12]:

$$P_i = \frac{C_{ii}}{\sum_j C_{ji}}, R_i = \frac{C_{ii}}{\sum_j C_{ij}}$$

Since precision and recall are calculated per class i, we average the precision and recall across all classes to obtain and report single macro-averaged precision and recall metrics for each model:

$$P = \sum_i P_i, R = \sum_i R_i$$

F_1-score is the harmonic mean of precision and recall and allows us to summarize the overall performance of our models. Since F_1 is based on precision and recall, it also calculated on a per-class basis but can be macro-averaged over all classes by computing the F_1 score using the macro-averaged precision and recall. This macro-averaged F_1 score is what we report for our models:

$$F_1 = 2\frac{PR}{P+R}$$

Accuracy is a simple metric that is the fraction of correctly classified instances over all instances. While accuracy can be a misleading metric, it gives a high-level overview of how our models are performing and allows us to compare our model with other papers that primarily use accuracy as their evaluation metric.

The performance of each of our three architectures on the validation set can be seen in Table 1. Precision, recall, and F_1 scores reported are macro-averaged across all classes. Our 2D CNN model obtains a 0.37 F_1 score on classifying single dance images. Our combined 2D CNN + LSTM model achieves similar performance with a 0.36 F_1 score on classifying sets of image frames. This suggests that adding the LSTM to the architecture allowed the model to capture temporal information from the image frames as we had hoped since the model performs similarly to the 2D CNN on a more difficult task of classifying image sets. Our best performing model, the 3D CNN obtains an F_1 score of 0.41 on classifying image sets. It seems that convolving over the image set dimension in an early fusion approach captures the temporal dimension better than individually generating feature maps for frames and then feeding them sequentially through an LSTM in a later fusion approach.

Table 1: Performances on Validation Set

Model	Precision	Recall	F_1	Acc
2D CNN	0.38	0.37	0.37	0.36
2D CNN + LSTM	0.39	0.37	0.36	0.36
3D CNN	0.43	0.42	0.41	0.42

Additionally, we reserved the test dataset for evaluation on our best model, the 3D CNN architecture. We were able to obtain an F_1 score of 0.40 on the test set, similar to our performance on the validation set.

Table 2: Performance on Test Set

Model	Precision	Recall	F_1	Acc
3D CNN	0.41	0.39	0.40	0.41

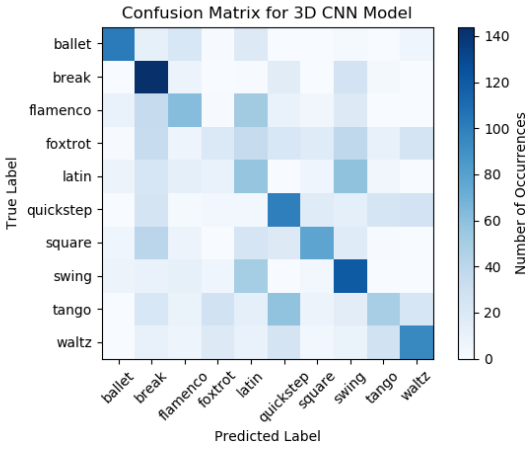


Figure 6: Confusion Matrix of 3D CNN Model [3]

Figure 7: Examples of Ballet

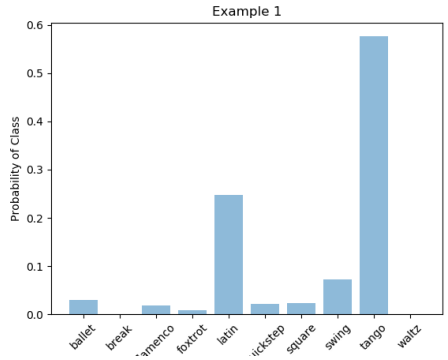
We also provide a visualization of the confusion matrix for our 3D CNN architecture on the validation set, as seen in Figure 6. Overall, the model does relatively better on classifying the correct dance over other dances, with the exception of Foxtrot. However, there are some dances commonly misclassified as others. For example, Flamenco is commonly misclassified as Latin which intuitively makes sense as Flamenco is a Spanish dance and shares many elements with other Latin dances. Another commonly mislabeled dance is that Tango is often mislabeled as Quickstep. This also makes sense as both are ballroom dances and contain many similar poses, as can be seen in Figure 10.

5.2. Error Analysis

As seen from our confusion matrix in Figure 6, most of our errors come from image sets of dances that are misclassified as different dances with similar poses or styles. One example of this can be seen in Figure 8, where a set of Latin dance images are misclassified as Tango. Upon inspection of the sample images, the images display long, outstretched poses, a key characteristic of Tango

style. While certain variations in Latin dances contain similar poses, it is understandable as to why the classifier may have mislabeled this set of images. This is a fitting example to demonstrate one of the challenges of classifying dance, and more generally, motion from images: many different dances contain similar poses or variations that can make it difficult to distinguish without additional context. Nevertheless our model does show evidence of learning key characteristics of each dance, as shown by the extremely low probabilities of every other class except Latin and Tango. This indicates that our model is learning, but it is having difficulties distinguishing between the last 2 or 3 candidate classes.

(a) 4 Frames of Latin from Validation Set



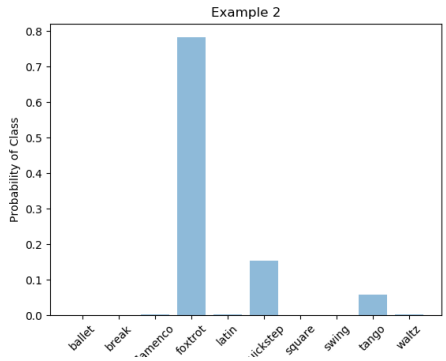
(b) Softmax Probabilities

Figure 8: Misclassified Example from Validation

On the other hand, our model was also able to capture many of the unique properties of each dance form. In the example in Figure 9, the model was able to correctly classify the image set as belonging to the Foxtrot dance with a high probability. Foxtrot's defining characteristics are its unique step pattern of slow-slow-quick-quick which can be seen from the sample image sequence as the lead appears to be in similar positions followed by sudden changes in pose. This demonstrates the potential for CNN models to capture the unique properties of specific dance and motion styles.

As seen in Table 3, there is a significant difference in the

(a) 4 Frames of Foxtrot from Validation Set



(b) Softmax Probabilities

Figure 9: Correctly Classified Example from Validation

performance of our best model across each class. Most notable is the relatively high success of Ballet classification, which we attribute to the very distinctive characteristics of the dance. The dance itself is typically performed in unique attire that is not present in other dance styles such as Waltz or Tango. Additionally, the atmosphere in which Ballet is performed is distinctive, oftentimes appearing on a theatrical stage with a extremely dark background, as seen in Figure 7.

Table 3: Individual Classes in Validation Set, 3D CNN

Class	Precision	Recall	F1
Ballet	0.76	0.63	0.68
Break	0.40	0.73	0.52
Flamenco	0.41	0.32	0.36
Foxtrot	0.23	0.09	0.13
Latin	0.22	0.31	0.26
Quickstep	0.40	0.47	0.43
Square	0.56	0.40	0.47
Swing	0.38	0.57	0.45
Tango	0.41	0.22	0.29
Waltz	0.54	0.46	0.50

Another thing worthy of note is the difficulty of drawing

clear lines between dances, especially when captured in video form. Unlike simple image classification with very defined categories such as *dog* or *cat*, dance styles are constantly evolving forms of human motion. As a result, different dance styles can oftentimes overlap as a result of dancers adapting a different dance move into a new style. This is what most likely led to the lower performances in some classes such as Latin or Tango, where they each contain strong aspects of the other.

5.3. Overfitting

For each of the models previously described, we also implemented several measures to prevent overfitting of the training set, which we noticed happened quickly with the chosen models. To combat this, we used a combination of dropout, regularization, and data augmentation. In each model, we applied dropout after each max pooling layer with a keep probability of 0.7 or 0.8. In addition to dropout, we also applied L2 regularization with a penalty factor of 0.1 to try and increase the validation accuracy. Finally, we also used data augmentation to try and diversify the training set a bit to help generalize better. To do so, we performed horizontal flips on all image frames in the training set and included them in the random batch sampling during training. We did not further augment by doing vertical flips or cropping due to the nature of the image frames. We reasoned that an upside down image frame would not capture generalizable information. Cropping would not make sense because in most videos, the figures do not take up a majority of the recorded space. Cropping has a high chance of losing valuable information (e.g. moving human figures) in this case.

Even with all the techniques discussed above, we were not able to significantly decrease the gap between training and validation accuracies. We hypothesize that this is due to the amount of training data available, and after a certain threshold, the model is unable to extract the minute differences between very similar classes. Figure 10 illustrates 3 classes present in our classification along with an example frame from each. Even to a human observer, it would be difficult to distinguish these based on a single frame or even several frames drawn consecutively. Across our models, our training accuracy peaked around 0.8 along with the validation scores listed earlier.

5.4. Hyperparameters

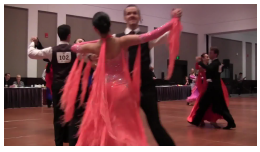
For each of the models described, we also performed an informal, random hyperparameter search to test the effects on performance. The most notable change came from learning rate, where the default 0.001 in Tensorflow's Adam Optimizer was several magnitudes too big [1]. We found that a learning rate of 1e-6 led to a steady loss decrease and convergence to local optima. Learning rates any bigger than this led to extremely noisy convergence, if at all, or a constant oscillation around the local optimum.

Additionally, we scaled the batch size in accordance to the model so that we could maximize the amount of resources available. This meant a batch size of 128 for the single image classifier with each image being 216 x 384. When dealing with 5 dimensional Tensors however, we had to significantly reduce the batch size to accommodate for the increase in parameters and image sets per example. Depending on the image set size for each example, our batch size for our 3D model was between 4 and 8.

We also looked at the possibility of image size possibly affecting the performance of the model. In addition to testing the original image size, we tested two other down-sampled images, detailed earlier. We found that there was no significant change in improvement across input frame size, and because of this, a lower input size would allow us to increase the image set size and batch size during training and testing.

For our models that required multiple frames to be extracted from a single video, we also had to tune the number of frames extracted and the number of frame skipped between each frame. Our best performing models ultimately landed on a balance of 12 image frames per video, with the start point randomly chosen such that 12 image frames could be extracted. Additionally, we also set the number of skip frames to be 8, so for every frame selected, the next 8 would be skipped. Finding a balance between these two hyper-parameters is crucial to balancing computational resources and important classification information. With unlimited computing resources, it is possible to incorporate every single frame, but here we try to extract a limited number of meaningful frames in the most efficient manner.

(a) Waltz (b) Tango



(c) Quickstep

Figure 10: Example Frames

6. Conclusion and Future Work

We evaluated the performance of 3 different models on the task of dance style classification, and we discovered that a 3D CNN yields the highest performance out of the 3 with an accuracy of 41% on the final test set. While running our models, we discovered the significant gap between training and validation accuracies, despite the use of several proven measures to combat overfitting. We suspect that this is due to the lack of breadth in the training dataset, and given either more data or more diverse classes, the model would most likely improve performance. A possible area where additional training data could be drawn from is YouTube using a platform provided by Google Research [2].

Additionally, it would be interesting to explore the use of large image sets in the 3D CNN and see whether or not a larger number of stacked image frames would better capture motion as it passes through the network. We also suspect that each of our models relies heavily on the background and setting present in the video as opposed to mostly on the human motion occurring. Due to this, future models could be augmented with a preprocessing layer that attempts to extract human motion specifically and deemphasize other aspects in a better measure of human motion understanding.

A final area of interest would be the use of additional information in the classification of dance styles. The inclusion of another feature such as music with the dance video may allow a deep learning model to learn the features of audio that correspond to a certain dance, and a ensemble of image and audio classification would be interesting to explore.

7. Contributions

The authors worked together to jointly develop and implement the models discussed in this paper. Henry Lin preprocessed the dataset, monitored model training, created the visualizations seen in this paper, and wrote the dataset section. Christopher Vo conducted the literature review and wrote the introduction, related works, and methods sections of this paper. Both authors contributed equally to the rest of the writing.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 7

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 8

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013. 5

[4] Daniel Castro, Steven Hickson, Patsorn Sangkloy, Bhavishya Mittal, Sean Dai, James Hays, and Irfan Essa. Let's dance: Learning from online dance videos. *arXiv preprint arXiv:1801.07388*, 2018. 1, 2

[5] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015. 2, 4

[6] Yin Fan, Xiangju Lu, Dian Li, and Yuanliu Liu. Video-based emotion recognition using cnn-rnn and c3d hybrid networks. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 445–450. ACM, 2016. 1, 2, 4

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 3

[8] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2, 4

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional

neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 3

[10] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4580–4584. IEEE, 2015. 1, 2, 4

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 3

[12] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009. 4

[13] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 412–428. Springer, 2016. 2

[14] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 2