CS230 Project: AI Choreographer

Songchun (Ray) Xie, Xiaoxia Feng, Theo Kanell
Stanford University
rayxie11@stanford.edu, colca7@stanford.edu, tkanell@stanford.edu

***Abstract*—This project presents the effort to achieve two goals in the process of making an AI choreographer. The first goal is to classify dance videos into correct dance genres. Each frame of the video is first fed into a feature extractor then to a custom built RNN model. Hyperband algorithm is leveraged to auto-tune hyperparameters in order to achieve the highest accuracy on the test set. The second goal is to predict a single frame of dancing given a series of past frames. A Conv-LSTM model is explored. The trained model shows capabilities in correctly predicting dance motion even in early stage iterations.**

I. Introduction

Thanks to the versatility of AI, Machine Learning and Neural Network, technology has advanced rapidly in many different areas such as self-driving cars, speech recognition systems, advertisement recommendation and etc. Some AI systems has even proved themselves to be highly innovative and creative. AI-generated artwork has won first prize in art competitions [1]. AI-composed music has helped musicians and artists to get started in new projects [2]. To further explore the capabilities of AI, the team has proposed to experiment on a different form of creative art: dance.

This project has two objectives: 1. develop a deep neural network that can categorize dance videos into different styles of choreograph correctly and 2. develop a generative neural network that can predict the next possible frame given a series of frames.

The first neural network can be used as a classification tool for mass video data generation. It can also facilitate downstream creative AIs such as choreography generation. In later sections, this model is referred as the **Classification Model**. The Classification Model employs a recurrent neural network (RNN) called bi-directional long-short term memory (LSTM) to output the softmax prediction for the possibilities of each choreograph style.

The second neural network for predicting next possible frame is more closely related to an "AI Choreographer". This model is referred as **Prediction Model**. The Prediction Model employs Conv-LSTM layers. These layers are LSTM layers with enhanced by a convolution recurrent cell. The 1 dimensional output of the convolution cell is used as input of the LSTM layer.

Both objectives in this project utilize an online public data set [3] of labeled short dance videos. There are a total of 16 different styles of choreographs, ranging from ballet and cha to sumba and waltz.

II. Related Work

A. Previous Motion Classification Work

Choreograph style classification can be abstracted into a more generic topic: human body motion classification. There are mainly two different ways to approach this problem: using videos and images or the time evolution of each human joint.

For motion classification using images and videos, extracting human motion features is most important. One prevalent method is to use transfer learning [4]: pass raw images or videos through a well established neural network (such as InceptionV3 [5]) to get the second to last layer output and use those features as the input for the new neural network. Since these neural networks have been well-trained and fine tuned with huge amounts of data, the features extracted depicts the image accurately.

As for specific dance related research, there are a number of attempts to build neural networks that classify or generate different choreographs. Even before the bloom of Machine Learning and AI, choreography planning using computer software was already developed. In 1968, Merce Cunningham envisioned the design of a computer technology that would enable 3 dimensional figures to be displayed on a computer screen. This led to the invention of LifeForms, a computer choreographic system [6]. For automated choreograph generation, DanceNet [7] is a successful attempt in generating dance moves from videos. This neural net employed Variational Autoencoder, LSTM and Mixture Density Network to generate silhouette choreographs.

For the Classification Model, the team draws inspiration from motion classification using images and videos. The team decided to approach the task using transfer learning as described above.

B. Previous Motion Prediction Work

The Prediction Model in this project leverages techniques and deep learning models from nowcasting [14] convective precipitation. Nowcasting technique was originally developed to apply in the field of weather forecasting. It gives precise and timely prediction of rainfall intensity in local region over short period of time in hours. Deep neural network has helped advancing nowcasting. RNN models provide new solutions beyond traditional approaches like a longer-term numerical weather prediction (NWP) model. Earlier LSTM encoder-decoder framework [15] provides a general framework for sequence-to-sequence learning problems by training temporally concatenated LSTMs: one for the input sequence and another for the output sequence.

The Conv-LSTM model used in this project a machine learning approach for precipitation nowcasting [12]. Precipitation nowcasting is reformulated as a spatiotemporal sequence forecasting problem. It can be solved under the general sequence-to-sequence learning framework. A novel convolutional LSTM (Conv-LSTM) network is proposed in order to model the spatiotemporal relationships. By stacking multiple Conv-LSTM layers and forming an encoding-forecasting structure, an end-to-end trainable model can be built. The versatility of this model is shown by testing and evaluating on the Moving-MNIST Dataset.

In this project, the team formulates choreography generation as a sequence-to-sequence learning task. The team leverages the Conv-LSTM architecture model to generate frames as an approach to predict dance motion in next frame.

C. Novelty

The use of video data separates our approach from prior methods. The team believe that training data should be accessible as well as bountiful. Video examples of dancing can be downloaded from the internet with ease. Straightforward collection is also perk: anyone can pick up their phone and start recording dance videos. Joint coordinates, on the contrary, needs sophisticated data collection equipment that is difficult to come by.

The Classification model takes advantage of transfer learning in order to boost our overall accuracy. We also custom built the neural network to correctly classify for our final prediction. In addition, the Hyperband algorithm was utilized to tune the hyperparameters of RNN model in order to gain the best validation accuracy.

The Prediction Model takes advantage of a convolution LSTM architecture neural network to predict dance motion in next frame given a sequence of past frames. In order to simplify the model for quicker processing, the team converts RGB video frames to grayscale frames.

III. Dataset and Features

A. Dataset Description

As mentioned in Section I, there are 16 different labels of choreograph in the dataset: ballet, break, cha, flamenco, foxtrot, jive, latin, pasodoble, quickstep, rumba, samba, square, swing, tango, tap, and waltz. The data set contains a total of 1398 correctly labeled examples. Each example contains about 100-300 frames.

B. Classification Model

1) Data Preprocessing: Even though there are only 1398 examples, each example contains more than 100 frames. Some even have more than 300 frames. Treating all frames for each example directly as inputs would not only be computationally expensive, but also prone to high variance due to the small number of samples. The team thus decided to limit each example 30 frames. For example, a 300 frame ballet video can be cut into 10 separate example each containing 30 frames. In the end, there are a total of 13603 examples. Each example would be labeled as a one-hot vector of length 16. For example, a foxtrot example would be labeled as:

$$y_{foxtrot} = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T \quad (1)$$

2) Train, Dev, Test Splits: The ratio between train and test set is 10:1. During model parameter tuning using dev sets, the team didn't achieve better results. Thus, there is no dev test.

3) Feature Extraction: As mentioned in Section II, transfer learning is employed in order to extract accurate and high level features from each frame of the example. The team decided to use InceptionV3 as the feature extractor for each frame. After reading a single frame from the original data set, it is passed through InceptionV3. Features are extracted from the second to last layer which outputs a vector of length 2048. All frames in the video are passed through the neural network which then forms a new example of (30×2048) (30 frames each with 2048 features).

C. Prediction Model

1) Data Preprocessing: For the Prediction Model, data is treated differently and each video example is processed in the following steps. First each example is limited to a subset number of frames (10, 20, 30) for faster training speed. Then, each video frame converted from RGB to grayscale to reduce dimensions. Early training shows that this conversion helps accelerate training speed and convergence rate. The intuition behind this is that grayscale frames will retain the same spatiotemporal motion information without distraction from color pixels. The reduced dimensions also reduces the number of parameters to train, hence improves the training speed. After that, frame shifting is employed. Each example contains a fixed number of frames n after step a. These frames are used for input x and label y generation using frame shifting. For instance, x would contain frames 0 to $n-1$, and y would contain frames 1 to n.

```
x = data[:, 0 : data.shape[1] - 1, :, :]
y = data[:, 1 : data.shape[1], :, :]
```

2) Train, Dev, Test Splits: The ratio between train and test set is 9:1. We didn't use a dev set.

IV. Classification Model Description

Since the motion of dancing is contingent upon time, the team decided to use a RNN as the main framework for classification task. To fully understand a single movement in a choreograph, it is better for the neural network to take both past and future movements into account. This is the reason why a bi-directional LSTM layer is used [10]. Before the features of each frame is fed into the bi-directional LSTM layer, it needs to be normalized first to reduce bias and variance. Thus, a batch normalization layer is used. After the bi-directional LSTM layer, a dropout layer is employed so that the network doesn't rely too heavily on one feature [11]. Later, 2 fully connected layers with non-linear activation functions are used. After that, 1 layer with a linear activation is used before passing it into a softmax layer. The softmax

layer guarantees to output a vector with values summed to 1 as shown in Equation 2.

$$\sigma(\mathbf{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, ..., K \text{ and } \mathbf{z} = (z_1, ..., z_K) \in \mathcal{R}^K \tag{2}$$

For loss, the team has chosen categorical cross-entropy loss with an Adam optimizer. Figure 1 shows the RNN model's architecture.

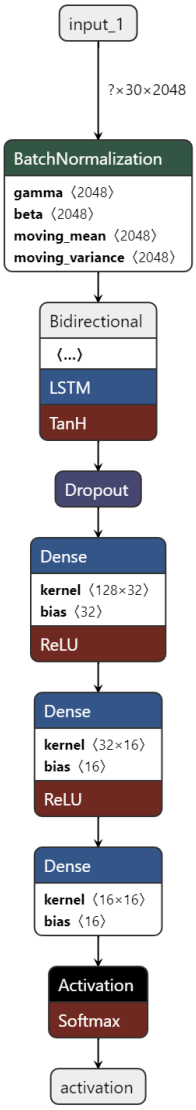

Fig. 1: RNN Model Architecture

V. Classification Model Experiments, Results, and Discussion

Table 1 shows the untuned hyperparameters used for the Classification Model.

With these hyperparameters, the team was able to achieve a high accuracy of 90.8%. Figure 2 shows the accuracy and loss for train and test sets with respect to number of epochs.

RNN Layer	Bi-directional LSTM
RNN Neurons	64
Dropout Rate	0.2
Dense Layer 1 Hidden Units	32
Dense Layer 2 Hidden Units	16
Epochs	20
Batch Size	128

TABLE I: Untuned Hyperparameters for Classification Model

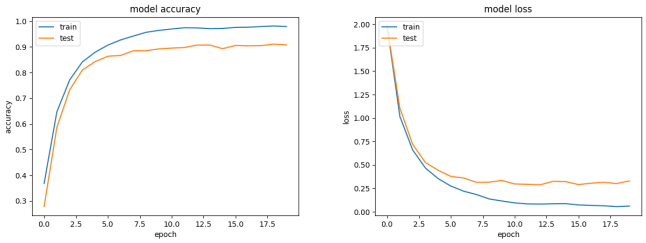

Fig. 2: Model Accuracy and Loss for Train and Test Sets for Classification Model

After attaining a good validation accuracy, we turned our focus on to hyperparameter tuning. We utilized the Keras tuner library to automate our hyperparemeter tuning. The parameters we focused on tuning were the number of neurons for the LSTM layer, the number of neurons for fully connected layers, the learning rate for the Adam optimizer, and the activation functions for fully connected dense layers.

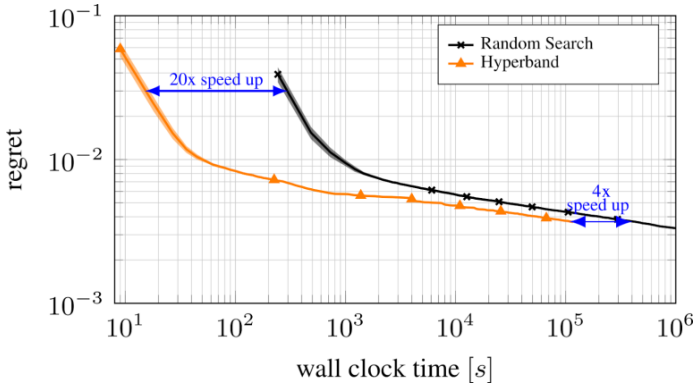

Fig. 3: Hyperband vs RandomSearch

The team employed the Hyperband algorithm to determine the best hyperparameters. From Figure 3, it is clear that Hyperband is much faster than Random Search at finding a "best" solution. This algorithm "uses adaptive resource allocation and early-stopping to quickly converge on a high-performing model" [8]. By using a technique called successive halving to prune lower performing sets, the algorithm is able to efficiently run just the top performing models. One unique aspect of Hyperband [9] is it takes full advantage of the resources available by treating different configurations as competitors in a bracket. It then compares each configuration and only continues to further epochs that are on the better half. This method significantly reduces run-time for our hyper-parameter

tuning which allows for further iteration in our model. The tuning runs $1+\log(\text{max\_epochs})$ per bracket where the number of brackets is dependent on system memory available.

RNN Layer	Bi-directional LSTM
RNN Neurons	448
Dropout Rate	0.2
Dense Layer 1 Hidden Units	320
Dense Layer 2 Hidden Units	160
Epochs	20
Batch Size	128

TABLE II: Tuned Hyperparameters for Classification Model

Table II shows the hyperparameters found by our Hyperband algorithm. In addition to the hyper-parameters shown, we changed the activation for the two fully connected dense layer to tanh, and used a learning rate of 0.0001 for our Adam optimizer.

After hyperparameter tuning, our model generated significant improvement which is shown in Figure 6. Our model reached 99% accuracy on our training data and a validation accuracy of 95.5% These results demonstrated a good trade off between bias and variance: near maximum accuracy for the training set and low variance. We also saw significant improvement when comparing the model loss. Using our tuned hyperparameters we were able to reduce our validation loss to 16.4% which is a significant improvement compared to the untuned model.

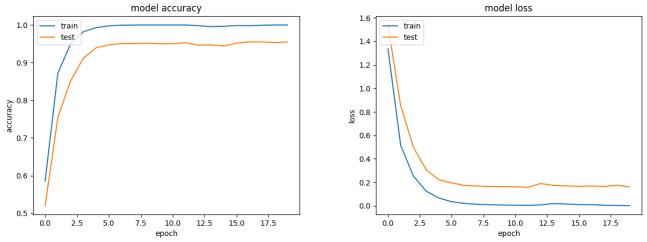

Fig. 4: Model Accuracy and Loss for Train and Test Sets for Classification Model Tuned

VI. Prediction Model Description

ConvLSTM2D [13] architecture combines gating of LSTM and 2D convolutions. ConvLSTM layers does a similar task to LSTM but instead of matrix multiplications, it does convolution operations and retains the input dimensions. After the images pass through the convolution layer, the result is flattened into 1D array and this will be the input to the LSTM layers with a set of features over time. ConvLSTM3D is similar and uses this layer to output a predict frame in its original dimensions. In our model, we constructed 3 ConvLSTM2D layers with batch normalization, and then a ConvLSTM3D layer for spatiotemporal outputs. Figure 4 shows the Conv-LSTM model architecture.

The input and output layer of the model share the same dimensions to retain the size of original frame, shape denoted by (num_samples, num_frames, image_height, image_width,

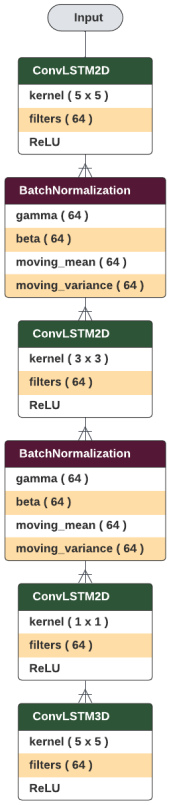

Fig. 5: Conv-LSTM Model Architecture

1). Once the model is trained for video generation, a sequence of single frames is passed to the model for prediction. The output frames are concatenated as the output prediction video. For loss, the team has chosen binary cross-entropy loss with an Adam optimizer.

VII. Prediction Model Experiments, Results and Discussion

A. Experiment With RGB Frames

In the preliminary model, the original colored frames were used to generate the dataset. Categorical cross-entropy loss was used for the model. However, the model could not converge, and validation loss also exploded. The team concluded that the colored pixels complicated the learning task, and a more complex model or different loss is required for training.

B. Experiment With Grayscale Frames

With the observation from the preliminary model, the team improved the model with data preprocessing. The frames were converted to grayscale before being input to the model leading to successful convergence. Accordingly, model loss is adjusted from categorical cross-entropy to binary cross-entropy. The output Conv-LTSM3D filters were adjusted for the grayscale output.

The grayscale images allows the model to be successfully trained, but exponentially higher processing power is needed

compared to the classification model. The previous classification model takes an image size (720×1080), and the Conv-LSTM model is limited to an image size of (60×90). With the memory and CPU constraints, we reached 0.4886 model loss after 10 epochs.

C. Next Frame Prediction and Video Generation

After the model was trained, we split the original video sample frames into two segments. The front segment is passed to the model to generate predicted frames, and the end segment is reserved for validation. We can convert the predicted frames to a gif and easily compare the prediction and original video clips.

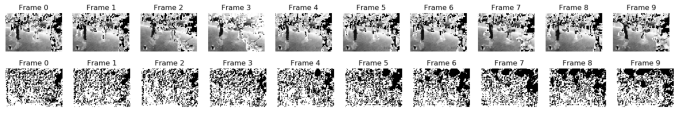

Fig. 6: Original/Predicted Frames

Figure 6 shows a comparison of original frames and predicted frames. Our prediction results shows: 1. Motion trends were successfully predicted even with low accuracy model. 2. Better prediction result are normally seen from clips with less noisy background and centered dancers. For the purpose of an AI Choreographer, higher quality dataset will improve prediction result: an ideal training dataset will be solo dancers with a clear background. 3. Given more iterations and higher resource capacity to improve the model, like using custom loss, the model could achieve better performance.

VIII. Conclusion and Future Work

A. Classification Model Has Better Performance

Metric	Untuned	Tuned
Train Accuracy	97%	99%
Train Loss	0.05	0.01%
Test Accuracy	90.8%	95.5%
Test Loss	0.48	0.2

TABLE III: Results for Classification Model

Our model was still performing at a lower accuracy rate than the other well developed motion classification neural networks. While previous models utilized more detailed data, we hoped to reach a similar degree of accuracy with only image data as mentioned in Related Works. Using the Hyperband algorithm for hyperparameter tuning, we were able to improve our model. The tuning found that increasing the number of neurons at different layers improved model performance. The increased complexity of the model allowed for our network to better account for the factors necessary to determine the dance category. The tuned model reached a validation accuracy of 95.5% which is within half a percentage point of the previous work. This is very exciting as it demonstrates that the model almost matched the previous model with smaller set of training data.

B. Prediction Model Can Be Improved

Despite an untuned model and limited epochs, the Prediction Model shows some interesting and promising early. We have outlined several necessary steps to improve the prediction model.

1) Custom Loss Function: We used binary cross-entropy which is a popular loss function in machine learning. However, a different loss function such as distance between pixel values on the original and generated frame could produce better results.

2) Augmented Data and Auto Tune: With more computational resources, we can augment the data, similar to classification, using full frame sections. We could also leverage the Hyperband algorithm to tune the model for improved performance.

C. Future Work

This research is a precursor to an automated choreography generator which fits the team's vision. An encoder and decoder could be included in a generative neural network. In addition, more diverse training data can be obtained (training set to test set) such as dance videos to time evolution of human joint coordinates, dance videos to time evolution of specific dance movements and etc. Moreover, a cross-validation method can be employed where the classification neural network can be used to as validation method to check whether the generative neural network has generated the correct style of choreography. Or an ensemble of deep learning frameworks leveraging the video prediction model for general motion predictions.

IX. Code Availability

This neural network is available for download at: https://github.com/rayxie11/cs230_project.

X. Contributions

Ray Xie: Data collection and preprocessing, research on related work, design and build neural network architecture

Xiaoxia Feng: Data preprocessing, AWS setup, design and build neural network, model debugging

Theo Kanell: Data research with social media sources, data preprocessing, hyperparameter tuning

References

[1] K. Roose, "An a.i.-generated picture won an art prize. artists aren't happy.," The New York Times, 02-Sep-2022. [Online]. Available: https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html. [Accessed: 26-Nov-2022].

[2] "Ai Music Generator - SOUNDRAW." [Online]. Available: https://soundraw.io/. [Accessed: 27-Nov-2022].

[3] D. Castro, S. Hickson, P. Sangkloy, B. Mittal, S. Dai, J. Hays, and I. Essa, "Let's dance: Learning from online dance videos," arXiv.org, 23-Jan-2018. [Online]. Available: https://arxiv.org/abs/1801.07388. [Accessed: 26-Nov-2022].

[4] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

[5] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[6] "Merce Cunningham + biped: Meet the master artist through one of his most important works," The Kennedy Center. [Online]. Available: https://www.kennedy-center.org/education/resources-for-educators/classroom-resources/media-and-interactives/media/dance/merce-cunningham–biped/. [Accessed: 01-Dec-2022].

[7] W. Zhuang, C. Wang, J. Chai, Y. Wang, M. Shao, and S. Xia, "Music2Dance: DanceNet for music-driven dance generation," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 18, no. 2, pp. 1–21, 2022.

[8] Li, Lisha, et al. 'Efficient Hyperparameter Optimization and Infinitely Many Armed Bandits'. CoRR, vol. abs/1603.06560, 2016, http://arxiv.org/abs/1603.06560.

[9] Introduction to the Keras Tuner: Tensorflow Core. TensorFlow. (n.d.). Retrieved December 6, 2022, from https://www.tensorflow.org/tutorials/keras/keras_tuner

[10] Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

[12] Shi, Xingjian & Chen, Zhourong & Wang, Hao & Yeung, Dit-Yan & Wong, Wai Kin & WOO, Wang-chun. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.

[13] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In NIPS, pages 3104–3112, 2014.

[14] Bližňák, V., Sokol, Z., & Zacharov, P. (2017). Nowcasting of deep convective clouds and heavy precipitation: Comparison study between NWP model simulation and extrapolation. Atmospheric Research, 184, 24–34. https://doi.org/10.1016/j.atmosres.2016.10.003

[15] Amogh Joshi. Next-Frame Video Prediction with Convolutional LSTMs. Keras. 2021. https://www.overleaf.com/project/63928c3d2c7eeb44e7cb235b