Deep Learning for Koopman–based Dynamic Movement Primitives

Tyler Han[1] and Carl Glen Henshaw, PhD[2]

ABSTRACT

The challenge of teaching robots to perform dexterous manipulation, dynamic locomotion, or whole–body manipulation from a small number of demonstrations is an important research field that has attracted interest from across the robotics community. In this work, we propose a novel approach by joining the theories of Koopman Operators and Dynamic Movement Primitives to Learning from Demonstration. Our approach, named Autoencoder Dynamic Mode Decomposition (aDMD), projects nonlinear dynamical systems into linear latent spaces such that a solution reproduces the desired complex motion. Use of an autoencoder in our approach enables generalizability and scalability, while the constraint to a linear system attains interpretability. Our results are comparable to the Extended Dynamic Mode Decomposition on the LASA Handwriting dataset but with training on only a small fractions of the letters.

I. INTRODUCTION

One facet of robotic learning entails training a robot to perform a task with a small number of demonstrations provided from an expert. Simple Learning from Demonstration (LfD) problems may be solved by the straightforward application of function approximation, *eg* trajectory examples provided by an operator can be approximated using splines and replayed as needed. However, this approach is suitable only for problems where the problem is highly constrained: with the robot working in the same area of the workspace, identical objects being manipulated across trials, and so on. As such, this type of technique does not readily allow changes in the task such as novel trajectories or variations in timing. Such techniques have found widespread use in factory automation, where these constraints can be enforced, but are less satisfactory in unstructured environments such as agriculture, disaster response, or domestic service.

Reinforcement Learning (RL) is an obvious technique to consider for LfD problems. Typically, RL requires long training times with many examples but an appropriate design of the architecture, along with the policy, loss, and/or reward, can result in significantly improved data efficiency[1], [2], [3]. Notable early work in RL for robotic motor learning was done by Schaal [4].

An alternative approach entails treating motions as an output of an underlying dynamical system. This approach is known Dynamic Movement Primitives (DMP) [5], [6], [7], [8] and provides substantial motivation for the framing of our work. DMPs formulate the motion generation for an LfD problem as a basin attractor system, normally designed by hand, and usually augmented with a learned forcing term. DMPs have been successfully used for a variety of LfD problems [9], [10]. A recent survey of the various mathematical formulations and adaptations can be found in [11].

In the field of dynamical systems, a burgeoning area of study is the data-driven identification of Koopman operators[12], [13], [14]. As opposed to typical differential equations descriptions of a dynamical system, which formulate the system as the evolution of a finite–length state variable, Koop-

[1]Tyler Han is an undergraduate intern at the Naval Center for Space Technology, U.S. Naval Research Laboratory, Washington DC, USA 20375 tyler.han@nrl.navy.mil

[2]Glen Henshaw is with the Robotics and Machine Learning Section, Naval Center for Space Technology, U.S. Naval Research Laboratory, Washington DC, USA 20375 glen.henshaw@nrl.navy.mil

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.

man theory formulates the dynamical system as the evolution of *observables*. The set of observables is the set of all scalar functions on the state, and hence the Koopman operator $\mathcal{K}$ is an infinite dimensional operator. However, importantly, the Koopman operator is a *linear* operator. With classic differential equations we accept nonlinearity in order to work with finite state vectors but with Koopman theory we work with a potentially infinite observation vector in order to have linear dynamics.

Although infinite–dimensional vectors are difficult to compute with, Koopman analysis suggests that in some cases, a system of interest can be either exactly or approximately represented with finite approximations to the Koopman operator [12]. Obviously, infinite–dimensional operators are difficult to store and compute but Koopman analysis suggests that in some cases, a system of interest can be either exactly or approximately represented with finite approximations to the Koopman operator [12]. This involves identifying a particular set of functions known as *observables* on the state of the system.

Classically, there are basis sets of functions that are known to be useful in the approximation of the Koopman operator; these include truncated monomial and polynomial expansions; transcendental functions; delay functions; and radial basis functions. However, the identification of appropriate observable functions for specific applications remains one of the fundamental challenges in Koopman theory. As a consequence, deep learning frameworks have attracted interest for approximating observable functions and the resultant latent space representations they induce [14].

Here, we join the the theories of DMPs and Koopman operators in a novel approach to robotic motion. As a preliminary experiment, we verify the approach using a handwriting dataset compiled by the Learning Algorithms and Systems Laboratory (LASA). We refer to our approach as aDMD in reference to the well–studied techniques Dynamic Mode Decomposition (DMD) and Extended Dynamic Mode Decomposition (eDMD).

Related Work

Notably, Lian and Jones [15] provide a rigorous framework for learning both the observation functions and the Koopman operator from data. They used Gaussian processes, a universal approximator, to learn observation functions and the resultant latent space, and demonstrated good results on the LASA dataset [15]. Lian and Jones make no claim that the learned observation functions are suitable for trajectories or tasks that are not in the training set, however, and demonstrated performance on individual characters. As a consequence, a disadvantage of this approach is the need to train a separate dynamical model for each character individually, including the observation functions, and retrain if new character strokes are desired. Here, we demonstrate equivalent performance to Lian and Jones with character stroke models using a fraction of the character strokes as training data, without requiring retraining for new characters.

II. Background

We assume that there exists an underlying dynamical system which dictates the flow of the states $x \in S$ of a system. Regardless of what information is realistically obtainable by the designer, x abstractly contains all the information needed to describe the system's instantaneous state. In the context of discrete systems, we denote these evolutions $x_{i+1} = f(x_i)$ where $f : S \to S$ represents the underlying dynamics, which in general are nonlinear.

As mentioned, Koopman theory formulates the dynamical system as the evolution of observables, which are scalar-valued functions of the state. This is typically described via operator theory as

$$\mathcal{K}g = g \circ f \tag{1}$$

$$\Rightarrow \quad \mathcal{K}g(x_i) = g(f(x_i)) = g(x_{i+1}) \tag{2}$$

where $g : S \to \mathbb{R}$ is an observable function. A comprehensive overview of Koopman operator theory is provided in the book by Kutz *et al* [12].

Dynamic Mode Decomposition (DMD) is one technique for determining a finite approximation to the Koopman operator. In DMD, data X, representing the state vector over time, is captured from the

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.

real system and the dynamics are approximated using a least-squares solution. Let $\mathbf{x}_i$ be the columns of $X \in \mathbb{R}^{n \times m}$. Denote $X_1 = \begin{bmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_{n-1} \end{bmatrix}$ and $X_2 = \begin{bmatrix} \mathbf{x}_2 & \dots & \mathbf{x}_n. \end{bmatrix}$ By assuming that the system is linear, one can solve the equation $X_2 \approx \tilde{A} X_1$ for $\tilde{A}$ via a pseudoinverse. This solution represents a least–squares fit to a linear dynamical system. eDMD generalizes this technique to the space of observables such that $Y_2 \approx \tilde{A}_e Y_1$ where $Y_1 = g(X_1)$, $Y_2 = g(X_2)$, and $g : \mathbb{R}^{n \times m} \to \mathbb{R}^{k \times m}$ represents a set of k observable functions $g_i : \mathbb{R}^n \to \mathbb{R}$ performed on each column of the input. Choosing an appropriate set of observable functions is in general a difficult problem. Approximating these functions using neural networks is one of the goals of this work.

III. Approach

A. Autoencoder Dynamic Mode Decomposition (aDMD)

Our model is largely adapted from [14], where the authors employ an autoencoder but use an auxiliary network in the identification of a Koopman operator. In contrast, our approach is to focus solely on the discovery of a set of observable functions for representing multiple trajectories as a discrete spectrum Koopman operator (see Fig. (1)). We further propose that the set of identified observables (encoder) is extensible to other "similar" trajectories not specifically trained on.

The process of aDMD is equivalent to eDMD by using latent representations from the autoencoder as the set of observables as a linear system. Equation (2) imposes requirements on the latent representation so that propagating a trajectory from an initial condition in latent space and using the decoder to return to delay space, we can compute the linear, prediction, and reconstruction losses. Linear loss (7) ensures that the latent space of the trajectory is indeed linear, as dictated by a set of linearizing observable functions. Prediction loss (8) ensures returning to delay space from latent space corresponds to the correct points in the trajectory. Reconstruction loss (9) is the standard loss as dictated by an autoencoder, requiring that data transformed by the encoder can be recovered

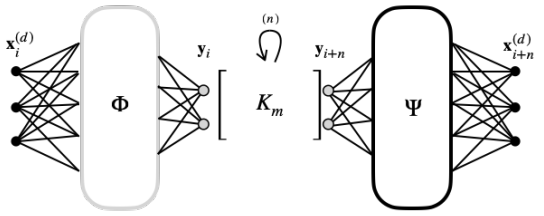

Fig. 1. Diagram of aDMD Architecture, composed of an encoder Φ and decoder Ψ surrounding an eDMD-computed recurrent state transition matrix, K_m.

with maximal accuracy. Finally, a regularization term is added to prevent overfitting.

Formally, let $\Phi : \mathbb{R}^{n_d} \to \mathbb{R}^m$ and $\Psi : \mathbb{R}^m \to \mathbb{R}^{n_d}$ represent the encoder and decoder networks of the autoencoder, respectively. Let $K : \mathbb{R}^m \to \mathbb{R}^m$ represent the approximated Koopman operator, which is calculated using $\Phi(\cdot) = \begin{bmatrix} \phi_1(\cdot) & \phi_2(\cdot) & \dots & \phi_m(\cdot) \end{bmatrix}^T$ where each $\phi_j : \mathbb{R}^{n_d} \to \mathbb{R}$ is an eDMD observable function. K_m is calculated via a pseudoinverse,

$$K_m = Y_2 Y_1^\dagger \tag{3}$$

where the latent state data matrices Y_1 and Y_2 are transformed snapshots of the data as in

$$Y_1 = \begin{bmatrix} | & | & & | \\ \Phi(\mathbf{x}_1^{(d)}) & \Phi(\mathbf{x}_2^{(d)}) & \dots & \Phi(\mathbf{x}_{N-1}^{(d)}) \\ | & | & & | \end{bmatrix} \tag{4}$$

$$Y_2 = \begin{bmatrix} | & | & & | \\ \Phi(\mathbf{x}_2^{(d)}) & \Phi(\mathbf{x}_3^{(d)}) & \dots & \Phi(\mathbf{x}_N^{(d)}) \\ | & | & & | \end{bmatrix}. \tag{5}$$

For a single trajectory, we use the loss function

$$\mathcal{L} = \mathcal{L}_{lin} + \alpha \mathcal{L}_{pred} + \mathcal{L}_{recon} + \beta \|\mathbf{W}\|_2^2 \tag{6}$$

$$\mathcal{L}_{lin} = \sum_{m=1}^{N} \|K^m \Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_{m+1})\|_{\text{MSE}} \tag{7}$$

$$\mathcal{L}_{pred} = \sum_{m=1}^{N} \|\Psi(K^m \Phi(\mathbf{x}_1)) - \mathbf{x}_{m+1}\|_{\text{MSE}} \tag{8}$$

$$\mathcal{L}_{recon} = \sum_{i=1}^{N} \|\Psi(\Phi(\mathbf{x}_i)) - \mathbf{x}_i\|_{\text{MSE}} \tag{9}$$

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.

where α and β are hyperparameters. Training over multiple trajectories, we represent the final loss function as the sum of the losses for each individual trajectories.

B. Delay Coordinates

Delay coordinates were introduced into DMD for discovering dynamics with standing waves [12] and, recently, linearity and forcing in strongly nonlinear (chaotic) dynamical systems [16]. They are a simple yet powerful way to compensate for data-sparse problems and allows the model to learn without overfitting the task space. In our case, the employment of delay coordinates can be seen as a way to "chunk" segments of a trajectory over time (Fig. (2)).

A coordinate $\mathbf{x}_i$ at time i is delayed once if instead of $\mathbf{x}_i$ we collect $\mathbf{x}_{i+1}$ at time i. We say that our data is augmented with d delay coordinates when each datapoint is delayed d times, each time augmenting the state vector with the resulting delayed state:

$$X^{(d)} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N-d} \\ \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_{N-d+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{d+1} & \mathbf{x}_{d+2} & \dots & \mathbf{x}_N \end{bmatrix} \tag{10}$$

For brevity, we will henceforth refer to the i'th column of the delayed data $X^{(d)}$ as $\mathbf{x}_i^{(d)} = \begin{bmatrix} \mathbf{x}_i^T & \mathbf{x}_{i+1}^T & \dots & \mathbf{x}_{i+d}^T \end{bmatrix}^T$. Note that the dimensionality of this vector is $n_d = n(d+1)$ and the duration (number of columns) of the delayed trajectory is now $N_d = N - d$.

C. Eigenmode Filtering

Even when using truncated SVD pseudoinverses, aDMD occasionally finds an "unstable" eigenmode — an eigenvalue of K greater than 1. We can filter out this eigenmode by subtracting it in latent space before returning to measurement space by determining the indexing set F whose corresponding eigenmodes are not unstable,

$$\tilde{X} = \Psi \left[\sum_{j=1}^{m} P_j^{(k)} - \sum_i P_i^{(k)} \right] = \Psi \left(\sum_{f \in F} P_f^{(k)} \right) \tag{11}$$

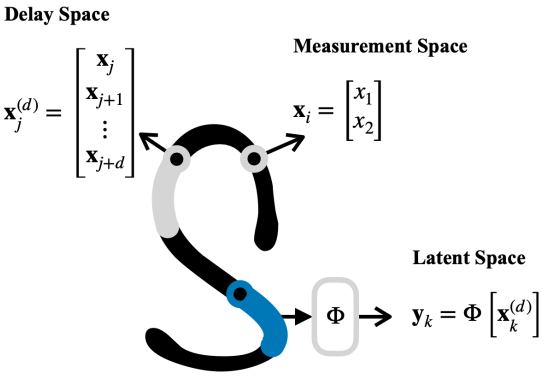

Fig. 2. Visualization of delay and latent space with respect to measurement space

D. Summary

In summary of our algorithm, we propagate the initial coordinate using the corresponding Koopman system:

1) Given the data X for a trajectory, obtain $X^{(d)}$ for the delay d with which the training data was augmented.
2) Identify K_m using equations (3), (5), and (11)
3) Let $\mathbf{x}_1^{(d)}$ be the first delay coordinate datapoint in the reconstruction
4) Other points in the reconstruction in delay coordinates are obtained using the rule
$$\mathbf{x}_i^{(d)} \approx \Psi(K_m^{i-1} \cdot \Phi(\mathbf{x}_1^{(d)}))$$
5) The full reconstruction in measurement space is given by the first n rows of the matrix whose ith column for $i \in [1, 2, \dots, N_d]$ is the reconstructed datapoint $\mathbf{x}_i^{(d)}$

IV. Results

We validate our approach on the LASA Handwriting dataset [17], which contains 26 handwritten, single-stroke "characters". Data augmentation was used to improve the robustness and generalizability of the aDMD. Augmenting the training set with random noise expands the training region and yields Koopman systems which can recreate letters even under perturbations of the initial conditions. This practice is also sometimes referred to as "motor babbling" [18].

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.

	aDMD	Polynomial eDMD ($n = 3$)	Polynomial eDMD ($n = 4$)
Average Error per Trajectory (Single Reconstruction)			
Prediction Error	0.0242399	0.1961263	0.0292222
Linear Error	0.0681664	1360.897	5407.519
Average Error per Trajectory (Noisy Reconstruction)			
Prediction Error	0.0561916	0.5516364	0.0705166
Linear Error	684.5577	4082.349	7291.988

TABLE I

COMPARISON OF ERRORS FOR ADMD AND POLYNOMIAL EDMD.

TABLE II

RECONSTRUCTION PLOTS COMPARING ADMD AND EDMD. ORIGINAL CHARACTERS ARE IN BLUE. RECONSTRUCTED TRAINING SET CHARACTERS ARE IN GREEN AND RECONSTRUCTED TEST SET EXAMPLES IN ORANGE.

Distribution Statement “A”: Approved for Public Release, Distribution Unlimited.

We evaluate the average prediction and linear error for single reconstructions, which take the datapoint $\mathbf{x}_1$ as the first point in the reconstruction, and the average prediction and linear error for noisy reconstructions, which take $\tilde{\mathbf{x}}_i \sim \mathcal{N}(\mathbf{x}_i, 0.05^2)$. The exact error equations are provided in the appendix.

Results are shown in Tables (I) and (II). aDMD reconstruction error is superior to 3–rd and 4–th order polynomial eDMD. Notable is aDMD's inherently compressive architecture. The number of effective states acted on by the autoencoder (delay space) is two times as large as the state in the learned linear system (latent space). Yet, the results are superior to polynomial eDMD. This provides evidence that the autoencoder achieves some level of compression perhaps adaptable to high–dimensional systems. In comparison, classical observable selection or dictionary methods (such as polynomial eDMD) typically explode or become intractable in high-dimensional systems [19].

V. Discussion & Further Work

We have shown that every example in the LASA Handwriting set can be described by observables identified by aDMD. The autoencoder $\Phi(\cdot)$ needs only one training example per trajectory and generalizes to the space of characters not yet seen. Test set characters are fit robustly, generally withstanding the same magnitude of perturbations on which the observables were trained.

We justify the framing of this approach as inspired by DMPs because, like DMPs, aDMD formulates the motion generation law for an LfD task as a dynamical system. Unlike other DMP approaches, however, aDMD allows us to automatically identify the appropriate dynamics directly from data, instead of hand–designing an attractor basin as is usually the case with DMPs.

For future work we intend to test on humanoid and quadruped locomotion and whole–body manipulation data, with the hypothesis that the movements of particular interest in those systems will exhibit analogous latent spaces, ultimately compressing and linearizing the dynamics and controls of interest to engineers. We also intend to investigate applying linear control theory within the linear latent space, with the intention of robustly handling system perturbations and external disturbances.

VI. Appendix

A. Results Error Calculation

For a given trajectory's data $\mathbf{x}_i \in \mathbb{R}^n$, errors for single reconstructions were calculated using

$$\mathcal{E}_{lin} = \sum_{m=1}^{N} \|K^m g(\mathbf{x}_1) - g(\mathbf{x}_{m+1})\|_{\text{MSE}} \quad (12)$$

$$\mathcal{E}_{pred} = \sum_{m=1}^{N} \|h(K^m g(\mathbf{x}_1)) - \mathbf{x}_{m+1}\|_{\text{MSE}} \quad (13)$$

where $g(\cdot)$ is the set of observables for the method and $h(\cdot)$ is the mapping back to measurement space (which includes taking the first n rows for aDMD).

For noisy reconstructions, 100 initial conditions were perturbed with a random variable sampled from a truncated multivariate normal distribution $\tilde{\mathbf{x}}_i \sim \mathcal{N}(\mathbf{x}_i, 0.05^2)$ and propagated into a trajectory.

$$\mathcal{E}_{n\ lin} = \frac{1}{100} \sum_{i=1}^{100} \sum_{m=1}^{N} \|K^m g(\tilde{\mathbf{x}}_1) - g(\mathbf{x}_{m+1})\|_{\text{MSE}} \quad (14)$$

$$\mathcal{E}_{n\ pred} = \frac{1}{100} \sum_{i=1}^{100} \sum_{m=1}^{N} \|h(K^m g(\tilde{\mathbf{x}}_1)) - \mathbf{x}_{m+1}\|_{\text{MSE}} \quad (15)$$

B. Training Details

When training against noise, $\mathbf{d}_i \sim \mathcal{N}(\mathbf{0}, 0.05^2)$ was added to every state $\mathbf{x}_i^{(d)}$ in every trajectory.

Hyperparameter values used in the results of this paper are: $\alpha = 100, \beta = 10^{-12}$ are used to tune the prediction and regularization losses, respectively. $m = 20$ is the dimension of the Koopman operator. $d = 20$ is the number of augmented delay coordinates. f, the activation function in the autoencoder hidden layers, was the standard ELU function; hid_width=20 and num_hidden=2 are the width and number of the autoencoder hidden layers, respectively. The model was trained using the Adam optimizer as implemented in the Julia programming language and Flux.jl library [20].

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.

REFERENCES

[1] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, pp. 278–287, 1999.

[2] A. D. Laud, *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.

[3] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[4] S. Schaal *et al.*, "Learning from demonstration," *Advances in neural information processing systems*, pp. 1040–1046, 1997.

[5] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics research. the eleventh international symposium*, pp. 561–572, Springer, 2005.

[6] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, IEEE, 2009.

[7] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[8] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[9] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural networks*, vol. 24, no. 5, pp. 493–500, 2011.

[10] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual learning from demonstration: adapting dynamic movement primitives for contact-rich insertion tasks," *arXiv preprint arXiv:2008.07682*, 2020.

[11] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *ArXiv*, vol. abs/2102.03861, 2021.

[12] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[13] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences of the United States of America*, 2016.

[14] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.

[15] Y. Lian and C. N. Jones, "Learning feature maps of the koopman operator: A subspace viewpoint," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 860–866, IEEE, 2019.

[16] S. L. Brunton, B. W. Brunton, J. L. Proctor, E. J. Kaiser, and N. Kutz, "Chaos as an intermittently forced linear system," *arXiv preprint arXiv:1608.05306v1*, 2016.

[17] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[18] R. Saegusa, G. Metta, G. Sandini, and S. Sakka, "Active motor babbling for sensorimotor learning," in *2008 IEEE International Conference on Robotics and Biomimetics*, pp. 794–799, IEEE, 2009.

[19] I. G. K. Matthew O. Williams, Clarence W. Rowley, "A kernel-based method for data-driven koopman spectral analysis," *Journal of Computational Dynamics*, vol. 2, no. 2, pp. 247–265, 2015.

[20] M. Innes, E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah, "Fashionable modelling with flux," *CoRR*, vol. abs/1811.01457, 2018.

Distribution Statement "A": Approved for Public Release, Distribution Unlimited.