Using Genetic Algorithms to Explore Germinal Center Mutation Strategies

Connor Frost
Computer Science
University of New Mexico
Albuquerque, United States
frostc@unm.edu

Craig Parry
Computer Science
University of New Mexico
Albuquerque, United States
parryc@unm.edu

Michael Adams
Computer Science
University of New Mexico
Albuquerque, United States
mikethebos@unm.edu

***Abstract*—Our project explores the recent innovations of Neural Cellular Automata's with relation to different datasets. Neural Cellular Automata (NCAs) leverage a deep learning network to both perceive and potentially react to its surrounding in a way that is similar to traditional CAs. By leveraging deep learning networks, this allows a non-discrete solution to CA rules and easier allows the NCAs to learn from well-established supervised datasets.**

This exploration with varying MNIST (Modified National Institute of Standards and Technology) datasets help us compare the results to state of the art results achieved by other models. This offers unique insights into the classification abilities of NCAs and the various problems that can be solved by this emerging technology.

I. Introduction

In traditional Cellular Automata (CA), there consists of a n-dimensional grid of cells where each cell consists of an on or off state. At each iteration, the grid is processed according to the ruleset and a new grid is generated. This iterative process of calculating the next iteration leads to interesting results. The most famous example of a 2D cellular automata is considered to be Conway's Game of Life, where a player creates an input and the evolutionary ruleset determines the output of the game. In neural cellular automata, the 3x3 convolution layer takes the place of a window for each grid cell. As seen in Fig 2, continuous-valued convolutions provide each cell the information about its neighborhood needed to update itself. A NCA's Convolutions provide two new abilities; They allow a cell's state to be in the range of continuous real numbers rather than discrete values, and any differentiable function can be used and will thus yield more sophisticated behavior than a regular CA. Theoretically, this has the ability to solve more complex problems.

In terms of the cost of computation, neural networks and cellular automata can both take advantage of parallelism. However, neural networks are said to be expensive since high end graphics processing units (GPUs) are generally needed to perform training in a reasonable amount of time. This is due to the complexity involved in performing back-propagation using gradient descent during supervised learning.

In our project we adapt and extend the code developed by the self-organizing NCA series [1], to accomplish these classification and generative approaches to data. We also look adversarial approaches and the overall cost of computation.

II. Datasets

A. Overview of Fashion MNIST

The Fashion MNIST dataset is considered a clothing and accessory dataset consisting of ten classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. The dataset consists of 60,000 training images, and 10,000 validation images, all of which are 28x28 pixels grayscale images. This is a MNIST-like dataset (similar to the dataset presented in the base paper) and is often used as a drop-in dataset. The background of the base images are typically solid black which helps us to differentiate between what is considered the "alive" cells and the dead cells.

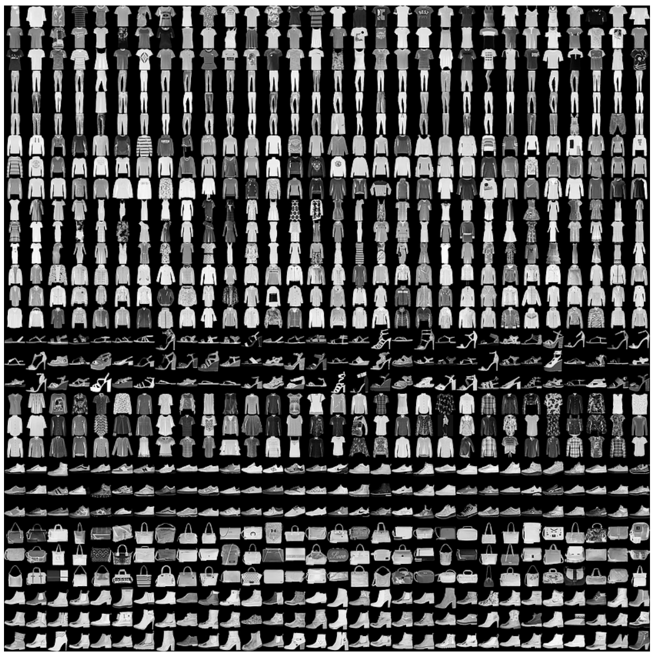


Fig. 1: Visualization of a subset of images from the Fashion MNIST dataset that is trained upon, in order from top to bottom: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot

The current state of the art results for the models is 96.91% achieved by the DARTS model [4], and where even the most simple of dense neural networks can achieve up to 91% [5]. Typically, model confusion arises in the classification of items such as t-shirts, pullovers, coats, and shirts where even human-level analysis can fail.

B. Overview of Kuzushiji MNIST

The Kuzushiji MNIST (KMNIST) dataset is composed of Japanese language characters consisting of the ten main classes of Hiragana: お, き, す, つ, な, は, ま, や, れ, を . [7] KMNIST is constructed similarly to Fashion MNIST with 28x28 grayscale images composing a training set of 60,000 examples, and a test set of 10,000 examples. It can be used as a drop-in replacement for MNIST and its color space allows for easy adaptation into NCA models.

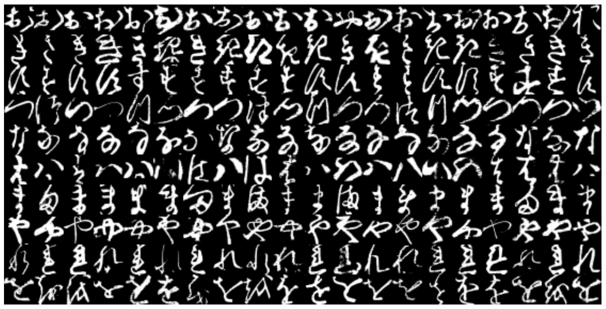


Fig. 2: Visualization of a subset of images from the KMNIST dataset that is trained upon. Each row represents a class of Hiragana; modern calligraphy is used on the left and ages as read from left to right.

State of the art models [8] achieve 99.76% with shake-shake regularization in order to minimize overfitting. In contrast to MNIST, Hiragana characters are mostly non-contiguous, meaning that they tend to feature islands of strokes separated by white space. This will prove significant to our NCA implementation since each input image is segmented into alive and dead cells.

III. Methods

A. Data Preprocessing

There is fundamentally an innate segmentation task of partitioning ”alive” cells (where the NCAs would be calculated) and the dead cells. In images, the alive cells would be defined by the pixels of the object subject to classification excluding the background. We do much of this data preprocessing by filtering out the background this is typically the darkest cells in the image, and manually calculate the alive cell mask.

Although there are specifically computer vision datasets for image segmentation, for simplicity we use both the Fashion MNIST and the KMNIST datasets. These are well known formats and we are able to differentiate the background from the foreground via filtering out the dark pixels with great accuracy. Unfortunately, it does seem that sometimes the clothing items such as the ones found in the Fashion MNIST dataset occasionally have dark pixels within the center. We suspect that this is due to the inclusion of graphic designs such as t-shirts where the design once converted to grayscale becomes dark enough to be filtered out with the background. Fortunately, we are able to still show model viability with regards to the dataset even with the included noise in the training data.

The input images for the neural cellular automata are normalized by rescaling the gray-scale pixel vaues between 0 and 1. By introducing a threshold of 0.1 to differentiate the background (consisting of dead-cells) from the foreground (which are considered light-cells).

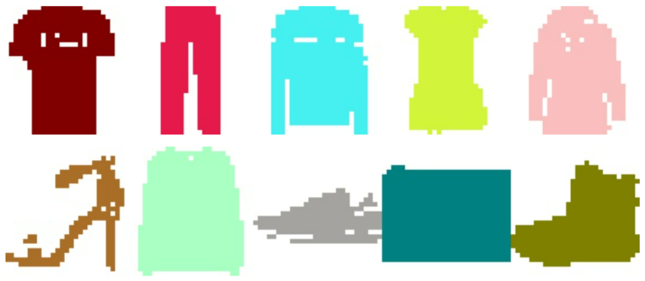


Fig. 3: Sample training set visualization of each clothing item with its corresponding colored-classification in the Fashion MNIST dataset

IV. Models

The models typically rely on two different model configurations that are used in conjunction: the perception model, and the decision model. The goal of the perception model is to emulate the perception of surrounding neural cellular automata. The goal of the decision model is to decide the update step and apply a stochastic update.

The following loss functions are considered to make back-propagation feasible for ”alive” cells:

- L2 Norm
- Cross-Entropy
- Sparse Categorical Cross-Entropy (after a softmax activation function)

A. Generative Model

Since the goal of the model is to regenerate an image that the neural cellular automata was trained upon, the output of the decision model must be the update state for that pixel. Since the process of regenerating the image from a single ”alive” cell, the number of alive and dead cells must be dynamic in nature for the complex interactions to grow. To accomodate this, at the end of each update step, new cells may be chosen around existing alive cells to allow for the growth of the generative model at the end of each update step.

- Input Layer: 28x28 color image.
- Perception Model - Layer 1: 2D Convolutional Neural Network with 3x3 kernel sobel operator (used for edge-detection) where padding is conserved.

- Decision Model - Layer 1: 2D Convolutional Neural Network, fully connected with ReLU activation function
- Decision Model - Layer 2: 2D Convolutional Neural Network with 16 outputs for update on each pixel.

B. Classification Model

The classification model is of similar form to the generative model with a perception and decision model. However, the decision model's goal becomes to output the correct classification of the pixel of the image and thus requires an activation function capable of producing a probability distribution function indicating the likelihood of each respective class. The perception model with the classification uses a trainable 2D convolutional neural network instead of the sobel operator in an attempt to better learn the features of the object.

- Input Layer: 28x28 grayscale image.
- Perception Model - Layer 1: Convolutional Neural Network with 3x3 kernel operator where padding is conserved
- Decision Model - Layer 1: Flatten output of perception model into 1D array
- Decision Model - Layer 2: 128 nodes, fully connected with ReLU activation function
- Decision Model - Layer 3: 10 nodes with classificationable activation function

The authors in [2] change the target image and class every 200 steps. This is done to ensure that each cell in the NCA is able to continuously classify rather than get stuck in a dormant state where the output classification never changes. In each of our training plots throughout the Results sections, this is visible as a discontinuity at step 200.

C. Adversarial Model

The adversarial begins with a pre-trained classification model and selects neural cellular automata to hijack. This is accomplished by modifying the training data to be intentionally corrupted and thus training the neural cellular automata using the "bad" data. Then, the original classification model has neural cellular automata repalaced with the adversarially trained neural cellular automata to maximize error.

V. Fashion MNIST Results

A. Fashion MNIST Classification Results

After training the classification model on the Fashion MNIST dataset, that has been preprocessed and normalized, for 100,000 epochs. The general accuracy of the model is calculated to be the following:

$$\frac{Total\_Correct\_CAs}{Total\_CAs} \tag{1}$$

The maximum final accuracy determined from the test dataset is calculated to be 80.81%.

Unfortunately, this metric is not the most informative metric as it doesn't account for agreement between cells. A secondary metric can be applied here as the agreement between cells.

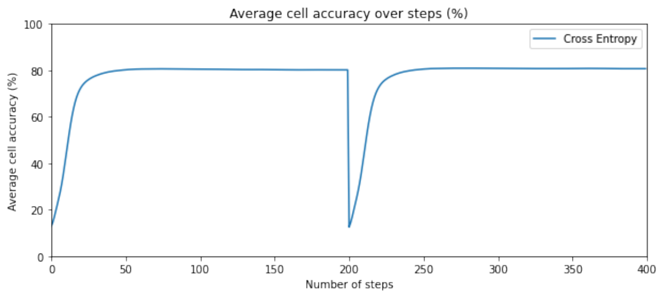


Fig. 4: Average cell accuracy for classification amongst pixels

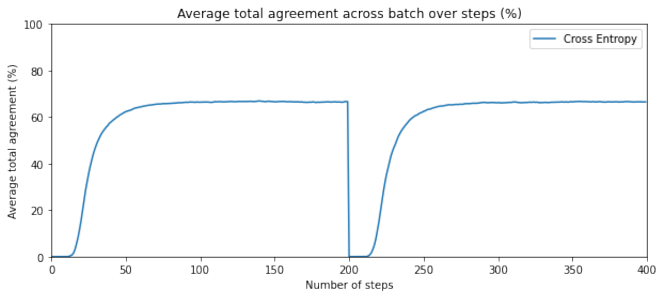


Fig. 5: Total agreement amongst pixels with respect to the classification that is being outputted for the Fashion MNIST dataset.

This can be computed as the ratio of samples where all the cells output the same label to the total number of samples.

By introducing this secondary metric, this attempts to help account for convergence of classifications amongst local regions with respect to each image. This metric becomes especially important with data that is not spatially connected. Due to the noise introduced in the pre-processing for the fashion MNIST dataset, there are occasionally "islands" of pixels that arrive to different classifications sometimes irrelevant to object of interest's classification.

Given the visualizations provided in fig. 5, the models become more proficient at classification as training progresses.

B. Fashion MNIST Generative Results

Using the generative model, we can train the neural cellular automata to regenerate an image that it was trained upon.

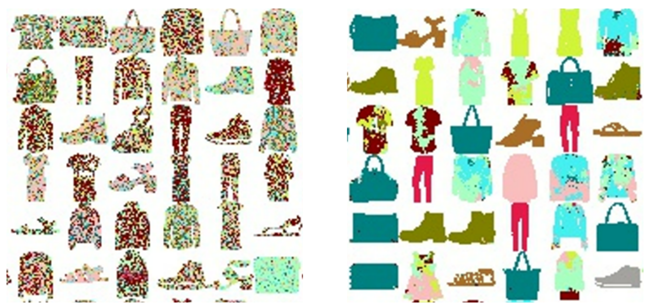
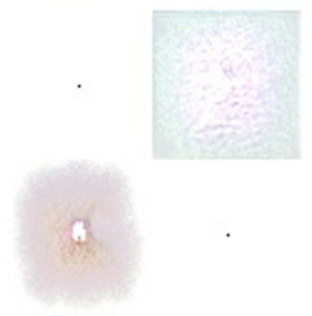


Fig. 6: Left: Visualization of training progress at epoch 100. Right: Visualization of training progress at epoch 100,000

This relies on an iterative approach to defining the alive and dead cells with each iteration. Upon each iteration, the cellular automata have been trained to generate the next iteration. However, the model is still limited by the designated alive cells.

Because the alive cells in the generative results are dynamic. The selection of ”new” cells to switch on must be performed after each iteration. The best way of going about this is to switch on cells surrounding other alive cells given a threshold. This is done at the end of each iteration to allow the model to grow relative to the activity of the model. There seems to be a trade-off by increasing the number of newly switched on cells increase the instability of the results. The slow and steady approach of the model leads the results to strictly converge to the image that it was trained upon.


(a) After 100 epochs

(b) After 500 epochs

(c) After 1000 epochs

(d) After 2000 epochs

(e) After 5000 epochs

(f) After 8000 epochs

TABLE I: Training progress over time in the generation of the images

From the information and the generated progress visualizations, the model seems to be able to recreate the original image in many different manners, but in the end after 8000 epochs, the neural cellular automata seem to be successful in the recreation.

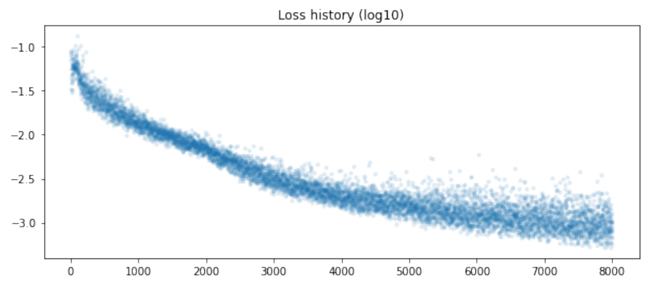


Fig. 7: Training loss of generation over time

C. Introducing Adversarial Training

By training an adversarial classifier model and selecting the most impactful pixels, the robustness of the model can be tested by utilizing these adversarially trained neural cellular automata with the rest of the originally trained classification model.

This is done in the following manner:

1) Train a classification model normally as described in earlier sections
2) Train an adversarial model with a distorted version of the dataset
3) Swap out 1% of pixels of the training model with the adversarial model
4) Test the results.

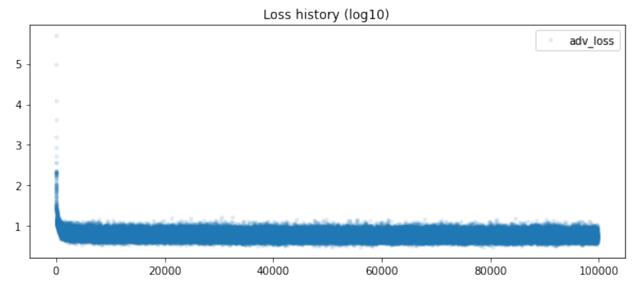


Fig. 8: Training loss of adversarially learning over time

The training results of this model is demonstrated above and demonstrate the impactfullness of the success that the adversarial models have in a naive sense where there is only a single adversarial response. It may be possible to train the original model with the adversarial nodes in mind such that it can learn to sometimes ignore nodes. This process can repeat itself many times to potentially generate more robust models.

VI. Kuzushiji-MNIST Results

A. Kuzushiji-MNIST Classification Results

The self classification model used on Fashion MNIST was also used to self-classify Japanese Hiragana character groups.

Average accuracy among every pixel CA is used here to measure overall accuracy.

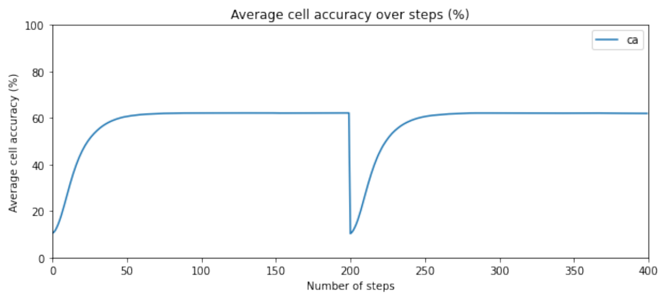


Fig. 9: Average accuracy among CA cells for Kuzushiji-MNIST dataset.

As seen in Figure 9, the model achieves 61.97% accuracy. However, as explained before for Fashion MNIST, agreement is a more important metric.

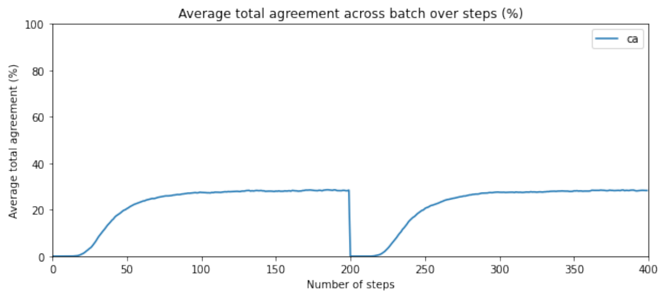


Fig. 10: Total agreement amongst pixels with respect to the classification that is being outputted for the Kuzushiji-MNIST dataset.

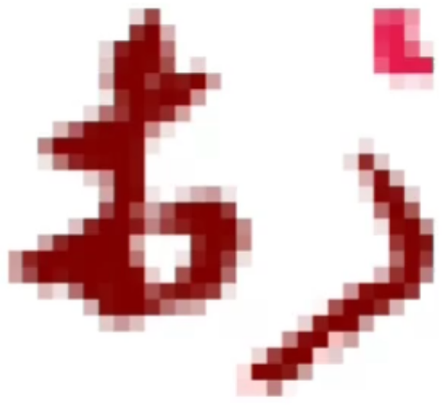


Fig. 11: An example hiragana character that has three islands of pixels.

Agreement is especially important with regards to Japanese Hiragana characters. Since a segmentation mask is used to prevent communication between islands of active cells, isolated calligraphic strokes present within some characters cannot communicate to converge on a consensus. Isolated strokes are common within Japanese calligraphy, so the final agreement of 28.26% seen in Figure 10 is not surprising. An example of a character with three islands can be seen in Figure 11.

Tu further verify the proficiency of self-classification on the KMNIST dataset, confusion matrices were used. Figure 12 shows the confusion matrix for a 100,000 step training run.

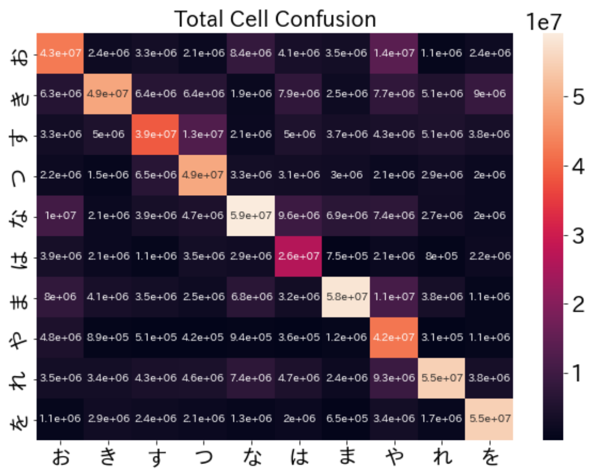


Fig. 12: Final training confusion matrix for each sample in the KMNIST test data set. Each row in a confusion matrix corresponds to the samples in an actual class while each column corresponds to the samples in a predicted class.

It can be seen that the model does fairly well at prediction once trained. Rows 2 and 5 (き, な, respectively) feature the most confusion as they have relatively high numbers of incorrect classifications in up to 4 other Hiragana character classes. The characters corresponding to rows 2 and 5 have a diverse number of strokes taken in all directions (vertical: up/down, horizontal: left/right), which could be a possible cause for the confusion exhibited, especially since some of the incorrect classifications are of images that have islands of cells like those seen in Figure 11.

B. Adversarial Experiments

Similarly to the experiments performed on the Fashion MNIST dataset, the adversarial model was trained on KMNIST, as well. Loss dropped rapidly within the first 2,000 training steps as seen before and were able to convince the original NCA to change its prediction to the れ Japanese character class. Figure 13 shows an example adversarial classification.

It can be seen that only two adversaries are needed in order to convince the pretrained NCA to change its class. This behavior can be seen for many other examples in the

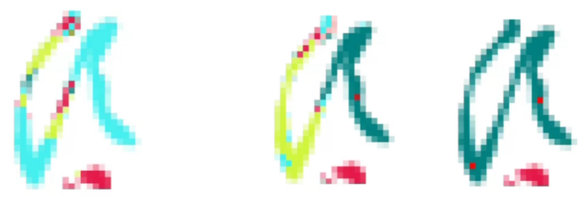


Fig. 13: Adversaries depicted by red dots. Left: Initial image iterated to a steady state by original NCA. Middle: Full effect of single adversary added to right stroke of character. Right: Subsequent effect of an additional adversary added to left stroke of character.

KMNIST test dataset. We predict that this is due to strategic placement of the adversaries. Observations of the animated classification process [1] showed that adversaries needed to be placed near stroke intersections in order to minimize the number of adversaries necessary. We suppose that this allows each adversary to communicate further along the shape, and thus, reinforcing other active adversaries.

VII. Discussion

This model still achieves decent results but has an element of visualization native to the model itself. While it may not be close to the state of the art results, it may be more ideal for those who are interested in models with an explainability / visualization capability over what you would typically see in black-box neural networks.

In addition, increasing the convolution size from 3x3 to 5x5 is completely possible to obtain better results as it increases the number of trainable parameters and also increases the scope for which results can be inferred upon. Unfortunately, this does lead to higher computation costs.

Currently, the generative model achieves the dynamic switching of cell states by turning on neighbors that go above a certain level of activity. This can be changed to become more dynamic such as moderating the total amount of new neighbors (by drastically reducing or increasing the limits), and by developing new dynamic rules for the switching. Currently, the model grows on pace with the addition of new cells and regeneration of the original image, however, increasing the instability may allow for the model to regenerate with some added variances.

Further changes to the NCA model may also improve KMNIST average accuracy and agreement among cells. Since islands of cells are commonly found within Hiragana characters, solely increasing the parameter space may not help. Instead, we propose future work look into an alternative to the alive/dead cell segmentation mask, or removing it altogether, to allow communication between islands.

[1]Videos of animations for several characters can be found in the code zip file accompanying this paper.

The KMNIST adversarial model brings further explainability to classification not found in neural networks. The ability to manually place adversaries and watch the NCA progress towards a prediction is valuable in a production setting and provides a peek into the black box mathematics of the convolutional layers. Adding adversaries on top of this and retraining the original NCA to counter them may yield confidence in a production deployment of NCAs if robustness develops.

VIII. Conclusion

An expanded analysis pf the ideas presented in [1], [2], and [3] were performed. Two new datasets were used as input into these existing models in order to evaluate how well NCAs can adapt to more complex data. While our classification results were subpar (80.81% and 61.97% for Fashion MNIST and Kuzushiji MNIST, respectively) when compared to state-of-the-art results using regular neural networks, they performed reasonably and have potential for architectural changes to increase performance.

However, NCAs offer increased explainability compared with regular neural networks that are considered to be black-box models without ability to be interpreted. NCAs performing classification or growth can be visually animated through all iterations, allowing computer scientists and domain experts to gain insight into what the model is doing, even though a NCA is still a black box model under the hood.

Future work should look at more complex image datasets, such as CIFAR-10 or Quick Draw, to see if NCAs can apply. Investigation into architectural and segmentation mask adjustments should accompany incorporation of new datasets to NCAs.

References

[1] Mordvintsev, Alexander, et al. “Growing Neural Cellular Automata.” Distill, 27 Aug. 2020, https://distill.pub/2020/growing-ca/.

[2] Randazzo, et al., ”Self-classifying MNIST Digits”, Distill, 2020.

[3] Randazzo, et al., ”Adversarial Reprogramming of Neural Cellular Automata”, Distill, 2021.

[4] M. S. Tanveer, M. U. K. Khan, C.-M. Kyung, ‘Fine-Tuning DARTS for Image Classification’, CoRR, . abs/2006.09042, 2020.

[5] TensorFlow, TensorFlow Docs, (2020), GitHub repository, https://github.com/tensorflow/docs

[6] H. Xiao, K. Rasul, R. Vollgraf, ‘Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms’, CoRR, . abs/1708.07747, 2017.

[7] Clanuwat, Tarin, et al. “Deep Learning for Classical Japanese Literature.” ArXiv.org, 3 Dec. 2018, https://arxiv.org/abs/1812.01718.

[8] Hysts. “Pytorch Implementation of Image Classification Models for CIFAR-10/CIFAR-100/Mnist/Fashionmnist/Kmnist” GitHub, github.com/hysts/pytorch_image_classification

[9] DeepLenin. “DeepLenin/Fashion-mnist_png: Fashion-Mnist Converted to PNG (with Script).” GitHub, https://github.com/DeepLenin/fashion-mnist$_{png}$.

Contribution Statements

A. Connor

- Worked on getting the classification up and running for Fashion MNIST
- Worked on getting the Adversarial training based on previous model runs

- Presented the work done on the project thus far on May 3rd, 2022.
- Worked on Fashion MNIST slides and data preparation slides.
- Wrote the model overviews section
- Helped write the introduction, abstract, and fashion MNIST results

B. Craig

- Worked on getting generative abilities with respect to the Fashion MNIST dataset
- Primary writer for the proposal
- Assisted with work of the adversarial training of Fashion MNIST
- Helped write data preparation section for the final paper. Contributed to the data preparation slides for the presentation.
- Wrote the Overview of Fashion MNIST section for the final paper
- Helped write the introduction, abstract, discussion, and fashion MNIST results.

C. Michael

- Wrote ML related sections of proposal
- Worked on running self classification for KMNIST
- Worked on running adversarial models for KMNIST
- Generated videos and text for KMNIST portion of presentation on May 3rd
- Wrote KMNIST results, bits of discussion, and complete conclusion