Garbage in, model out: Weight theft with just noise

Anonymous Authors[1]

Abstract

This paper explores the scenarios under which an attacker can claim that 'Noise and access to the softmax layer of the model is all you need' to steal the weights of a convolutional neural network whose architecture is already known. We were able to **achieve** 96% **test accuracy using the stolen MNIST** model and 82% accuracy using stolen KMNIST model learned using only i.i.d. Bernoulli noise inputs. We posit that this theft-susceptibility of the weights is indicative of the complexity of the dataset and propose a new metric that captures the same. The goal of this dissemination is to not just showcase how far knowing the architecture can take you in terms of model stealing, but to also draw attention to this rather idiosyncratic weight learnability aspects of CNNs spurred by i.i.d. noise input. We also disseminate some initial results obtained with using the Ising probability distribution in lieu of the i.i.d. Bernoulli distribution

1. Introduction

In this paper, we consider the fate of an adamant attacker who is adamant about only using noise as input to a convolutional neural network (CNN) whose architecture is known and whose weights are the target of theft. We assume that the attacker has earned access to the softmax layer and is not restricted in terms of the number of inputs to be used to carry out the attack.

At the outset, we'd like to emphasize that our goal in disseminating these results is not to convince the reader on the real-world validity of the attacker-scenario described above or to showcase a novel attack. This paper contains our initial explorations after a chance discovery that we could *populate* the weights of an MNIST-trained CNN model by just using noise as input into the framework described below.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Through a set of empirical experiments, which we are duly open sourcing to aid reproducibility, we seek to draw the attention of the community on the following two issues:

1. This risk of model weight theft clearly entails an interplay between the dataset as well as the architecture. Given a fixed architecture, can we use the level of susceptibility as a novel metric of complexity of the dataset?

2. Given the wide variations in success attained by varying the noise distribution, how do we formally characterize the relationship between the input noise distribution being used by the attacker and the true distribution of the data, while considering a specific CNN architecture? What aspects of the true data distribution are actually important for model extraction?

The rest of the paper is structured as follows:
In Section 2, we provide a brief literature survey of the related work. In Section 3, we describe the methodology used to carry out the attack. In Section 4, we cover the main results obtained and conclude the paper in Section 5.

2. Related work

The art form of *stealing* machine learning models has received a lot of attention in the recent years. In (Tramèr et al., 2016), the authors specifically targeted real-world ML-as-a-service (Ribeiro et al., 2015) platforms such as BigML and Amazon Machine Learning and demonstrated effective attacks that resulted in extraction of machine learning models with *near-perfect fidelity* for several popular model classes. In (Correia-Silva et al., 2018), the authors trained what they termed as a *copycat network* using *Non-Problem Domain* images and stolen labels to achieve impressive results in the three problems of facial expression, object, and crosswalk classification. This was followed by work on *Knockoff Nets* (Orekondy et al., 2018), where the authors demonstrated that by merely querying with random images sourced from an entirely different distribution than that of the black box target training data, one could not just train a well-performing knockoff but it was possible to achieve high accuracy even when the knockoff was constructed using a completely different architecture.

This work differs from the above works in that the attacker is adamant on only using *noise* images as querying inputs. Intriguingly enough, the state-of-the-art CNNs are not robust enough to provide a flat (uniform) softmax output (with weight $1/$*number-of-classes*) when we input non-input-domain noise at the input layer. This was been studied under two contexts. The first context was within the framework of *fooling images*. In (Nguyen et al., 2015), the authors showcased how to generate synthetic images that were noise-like and completely unrecognizable to the human-eye but ones that state-of-the-art CNNs classified as one of the training classes with 99.99% confidence. The second text was with regards to what the authors in (Goodfellow et al., 2014) stated to be *rubbish-class examples* . Here, they showcased that the high levels of confident mis-predictions exuded by state-of-the-art trained on MNIST and CIFAR-10 datasets in response to isotropic Gaussian noise inputs.
In this work, we focus on using Bernoulli noise-samples as inputs and using the softmax responses of the target model to siphon away the weights.

3. Methodology

3.1. Threat model

We propose a framework for model extraction without possession of samples from the true dataset which the model has been trained on or the purpose of the model other than the dimensionality of the input tensors as well as the ability to access the resulting class distribution from what is assumed to be a softmax activation given an input. We make the additional assumption that the architecture of the model to be extracted is known by the adversary. In our experiments, we assume that the input tensor is of dimension 28 by 28 and each pixel has values on the interval $[0, 1]$.

3.2. Victim model

The black box model which we attempt to extract, $F(\cdot)$, whose architecture is described in Table 3, is trained to convergence on a standard dataset for 12 epochs using the Adadelta optimizer with an initial learning rate of 1.0 and a minibatch size of 128 (Mni). From this point onward, this model is assumed to be a black box in which we have no access to the parameters of each layer.

3.3. Random stimulus response for model extraction

We procedurally generate a dataset of 'stimuli' comprised of 600000 28 by 28 binary tensors where each pixel is sampled from a Bernoulli distribution with a success probability parameter p. In other words, let each image $x^i_{rand} \in X_{rand} \subseteq \{0, 1\}^{28\times28}$ where $x^i_{rand,j,k} \sim \mathrm{Bern}(p)$ for $i \in \{1, ..., 600000\}$. We sample these tensors with probability parameters $p \in \{0.01, 0.11, ...0.91\}$, where each p is used to generate 10% of the data. We obtain predictions from the black box model for each randomly sampled example, $y^i_{rand} = F(x^i_{rand})$, which we refer to as 'responses.'

Table 1. Victim architecture as found in the MNIST example in the documentation for the Keras deep learning library.

LAYER TYPE	DIMENSIONS	ADDITIONAL INFORMATION
CONVOLUTIONAL	$32, 3 \times 3$	RELU
CONVOLUTIONAL	$64, 3 \times 3$	RELU
MAX POOLING	2×2	-
DROPOUT		RATE $= 0.25$
DENSE	128	RELU
DROPOUT	-	RATE $= 0.5$
DENSE	10	SOFTMAX

3.4. Extraction

We train a new model, $F_{extract}(\cdot)$, on the stimulus response pairs, $\{(x^i_{rand}, y^i_{rand})\}_{i=1}^{600000}$ pairs with no regularization and evaluate on the dataset originally used to train $F(\cdot)$. The architecture for this model is the same as $F(\cdot)$, except we remove the dropout layers to encourage overfitting. We train for 50 epochs using the Adadelta optimizer with an initial learning rate of 1.0 and a minibatch size of 128. Additionally, we acknowledge a significant class imbalance in the highest probability classes in the softmax vectors y_{rand}, so we remedy this by computing class weights according to the argmax of each softmax vector, and applying this re-weighting during the training of $F_{extract}(\cdot)$. We show the full extraction algorithm in Algorithm 1 and summarize it in Figure 1.

We evaluate our proposed framework on four datasets from the MNIST family of datasets with identical dimensions: MNIST, KMNIST, Fashion MNIST, and notMNIST (LeCun & Cortes, 2010; Clanuwat et al., 2018; Xiao et al., 2017; not).

3.5. Experiments with noise distributions

We evaluated the effect of sampling random data x^i_{rand} from different distributions on the performance of $F_{extract}(\cdot)$ on the MNIST validation set. We used the same training procedure as found in the previously described experiments with two exceptions: we sample only 60000 procedurally generated examples and we train $F_{extract}(\cdot)$ for only 10 epochs. We evaluated the use of the uniform distribution on the bounded interval $[0, 1]$, the standard normal distribution, the standard Gumbel distribution, the Bernoulli distribution with success parameter $p = 0.5$, and samples from an Ising model simulation with inverse temperature parameter $\beta \in [0.0, 0.1, ..., 0.9]$ and resulting values scaled to $\{0, 1\}$.

3.6. The Ising prior as a model of spatial correlation

The Ising prior is defined by the density (Taroni, 2015):

$$p(\mathbf{x}) = \frac{\exp\left[-\beta \sum_{ij \in E} (x_i x_j)\right]}{\sum_{\mathbf{x}} \exp\left[-\beta \sum_{ij \in E} (x_i x_j)\right]}; x_i \in \{-1, 1\}$$

Examples of images sampled from the Ising model can be found in Figure 6.

For this experiment, we evaluated the role of the inverse temeprature β parameter of the Ising sampler in training $F_{extract}(\cdot)$. We first partition the stimulus response pairs, (X_{Ising}, Y_{Ising}) into 10 subsets with 7000 examples each corresponding to the different β parameters used to generate the samples, where $(X_{Ising}, Y_{Ising}) = \bigcup_{\beta \in \{0.0, 0.1, ..., 0.9\}} \{(X_{Ising,\beta}, Y_{Ising,\beta})\}$. We train $F_{extract}(\cdot)$ for 10 epochs for each β and validate on the original dataset. We performed this experiment for MNIST, KMNIST, Fashion MNIST, and notMNIST and report the variation in performance over different values of β.

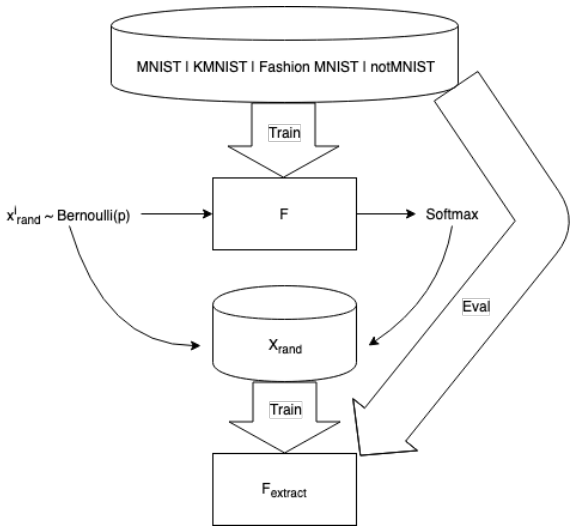

Figure 1. Overview of the model extraction algorithm.

4. Results

4.1. MNIST

We evaluate the efficacy of our framework by training $F(\cdot)$ on MNIST and going on to evaluate the performance of $F_{extract}(\cdot)$ on MNIST after extraction. We found that $F(\cdot)$ achieved a validation accuracy of 99.03% and $F_{extract}(\cdot)$ achieved a validation accuracy of 95.93%. The distribution of the argmax of Y_{rand} can be found in Figure 2. The most underrepresented class according to the argmax of Y_{rand} was class 6 represented by 198 out of 600000 random examples.

Algorithm 1 Stimulus response model extraction.

Input: data X_{train}, Y_{train}, X_{val}, Y_{val}
Initialize $F(\cdot)$.
Initialize *numRandomExamples* $= 600000$.
Initialize *dim* $= 28$.
Fit $F(X_{train})$, Y_{train}.
Evaluate $F(X_{val})$, Y_{val}.
for p in $\{0.01, 0.11, ..., 0.91\}$ **do**
  for q in $\{0, 1, ..., numRandomExamples/10\}$ **do**
    for j in $\{0, 1, ..., dim\text{-}1\}$ **do**
      for k in $\{0, 1, ..., dim\text{-}1\}$ **do**
        $x_{sample,j,k} \sim \text{Bern}(p)$
      end for
    end for
    $X_{rand} = X_{rand} \cup x_{sample}$
  end for
end for
Initialize $F_{extract}(\cdot)$.
for $i \in \{1, ..., |X_{rand}|\}$ **do**
  $y^i_{rand} = F_{extract}(x^i_{rand})$
end for
Compute class weights $CW_{Y\,rand}$ given Y_{rand}
Fit $F_{extract}(X_{rand})$, Y_{rand} with $CW_{Y\,rand}$.
Evaluate $F_{extract}(X_{val})$, Y_{val}.

4.2. KMNIST

Our experiments with KMNIST resulted in $F(\cdot)$ achieving a validation accuracy of 94.79% and $F_{extract}(\cdot)$ achieving a validation accuracy of 81.18%. Class 8 was found to be the class with the fewest representatives according to the argmax of Y_{rand}, which had 272 representative examples out of 600000.

4.3. Fashion MNIST

On the Fashion MNIST dataset, we found that $F(\cdot)$ achieved a validation accuracy of 92.16%, while $F_{extract}(\cdot)$ achieved a validation accuracy of 75.31%. For Fashion MNIST, the most underrepresented class according to the argmax of Y_{rand} was class 7 (sneaker) with only 12 out of 600000 random examples. Notably, the most common mispredictions according to Figure 3 were incorrectly predicting class 5 (sandal) when the ground truth is class 7 (sneaker) and predicting class 5 (sandal) when the ground truth is class 9 (ankle boot). $F_{extract}(\cdot)$ seems to predict the majority of examples from shoe-like classes to be of class 5 (sandal).

4.4. notMNIST

We found that the notMNIST dataset had a more uniform class distribution according to the argmax of Y_{rand} than the other datasets that we evaluated. The class with the fewest representatives in this sense was class 9 (the letter j)

Table 2. Performance using different noise distributions.

DISTRIBUTION	$F_{extract}(\cdot)$ VALIDATION ACCURACY
UNIFORM ($a = 0, b = 1$)	11.72%
STANDARD NORMAL ($\mu = 0, \sigma = 1$)	68.79%
STANDARD GUMBEL ($\mu = 0, \beta = 1$)	70.03%
BERNOULLI ($p = 0.5$)	76.58%
ISING ($\beta \in \{0.0, 0.1, ..., 0.9\}$)	98.02%

with 3950 out of 600000 examples. Despite this potential advantage, the extracted model $F_{extract}(\cdot)$ failed to generalize to the notMNIST validation set, achieving an accuracy of 10.47%, and as can be seen in Figure 3, $F_{extract}(\cdot)$ predicts class 5 (the letter e) in the vast majority of cases. In contrast, $F(\cdot)$ achieved a validation accuracy of 88.62%.

4.5. The performance of different noise distributions

In evaluating the effect of sampling from different distributions to construct X_{rand}, we found that among the uniform, standard normal, standard Gumbel, Bernoulli distributions, and the Ising model, samples from the Ising model attained the highest accuracy at 98.02% when evaluating $F_{extract}(\cdot)$ on the MNIST validation set. The results for each of the other distributions can be found in Figure 2. We postulate that this is due to the modelling of spatial correlations, which is a property which is lacking when sampling from the uniform, standard normal, standard Gumbel, and Bernoulli distributions, as the pixels are assumed to be i.i.d.

4.6. Extraction hardness resulting from data

We propose a measure of model extraction hardness resulting from the dataset which the original model is trained on as the ratio of the post-extraction validation accuracy (using $F_{extract}(\cdot)$) and the pre-extraction validation accuracy (using $F(\cdot)$) under our framework. We show that the resulting ratios are align with the mainstream intuition regarding the general relative learnability of MNIST, KMNIST, Fashion MNIST, and notMNIST. For MNIST, we found this ratio to be 0.9687, the ratio for KMNIST was 0.8564, for Fashion MNIST we found it to be 0.8171, and notMNIST achieved a ratio of 0.1181.

4.7. The role of modelling spatial correlation

We found that the loss and accuracy undergo 'phase transitions' as the value of β is varied. In Figure 4, we see that across datasets, the losses tend to be minimized around $\beta = 0.3$, however the behavior of larger values of β varies from dataset to dataset. We postulate that this is indicative of the different distributions of the amount of spatial correlation across each dataset. We also found that accuracy is maximized at $\beta = 0.4$ for MNIST, $\beta = 0.3$ for KMNIST and Fashion MNIST, and $\beta = 0.2$ for notMNIST, where the behavior here also varies as β increases from the optimal value. We show this in Figure 4.

Table 3. Performance on original dataset before and after extraction (measured on the validation set).

DATASET	PRE-EXTRACTION ACCURACY	POST-EXTRACTION ACCURACY
MNIST	99.03%	95.93%
KMNIST	94.79%	81.18%
FASHION MNIST	92.16%	75.31%
NOTMNIST	88.62%	10.47%

5. Conclusion and future work

In this paper, we demonstrated a framework for extracting model parameters by training a new model on random impulse response pairs gleaned from the softmax output of the victim neural network. We went on to demonstrate the variation in model extractability based on the dataset which the original model was trained on. Finally, we proposed our framework as a method for which relative dataset complexity can be measured.

5.1. Future work

This is a work in progress and we are currently working along the following three directions: In our experiments, pixels are notably i.i.d., whereas in real world settings, image data is comprised of pixels which are spatially correlated. In this vein, we intend to establish the relationship between the temperature of an Ising prior and the accuracy obtained by the stolen model. We will experiment with different architectures, specifically exploring the architecture unknown scenario where the attacker has a fixed plug-and-play swiss-army-knife architecture whose weights are learned by the noise and true-model softmax outputs. Additionally, we will explore methods for constructing X_{rand} which gives more uniform distributions over $\text{argmax}(Y_{rand})$ and evaluate the associated effect on the performance of $F_{\text{extract}}(\cdot)$.

References

Mnist cnn - keras documentation. https://keras.io/examples/mnist_cnn/. (Accessed on 05/20/2019).

notmnist — kaggle. https://www.kaggle.com/jwjohnson314/notmnist. (Accessed on 05/21/2019).

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature, 2018.

Correia-Silva, J. R., Berriel, R. F., Badue, C., de Souza, A. F., and Oliveira-Santos, T. Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

Orekondy, T., Schiele, B., and Fritz, M. Knockoff nets: Stealing functionality of black-box models. *arXiv preprint arXiv:1812.02766*, 2018.

Ribeiro, M., Grolinger, K., and Capretz, M. A. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 896–902. IEEE, 2015.

Taroni, A. Statistical physics: 90 years of the ising model. *Nature Physics*, 11(12):997, 2015.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 601–618, 2016.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

A. Additional figures

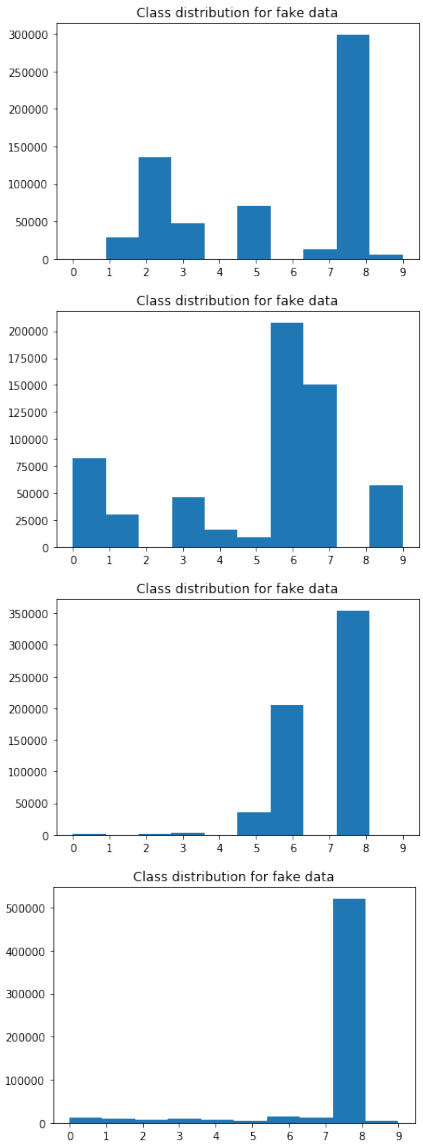

Figure 2. Distribution of classes given X_{rand}. From top to bottom: MNIST, KMNIST, Fashion MNIST, notMNIST.

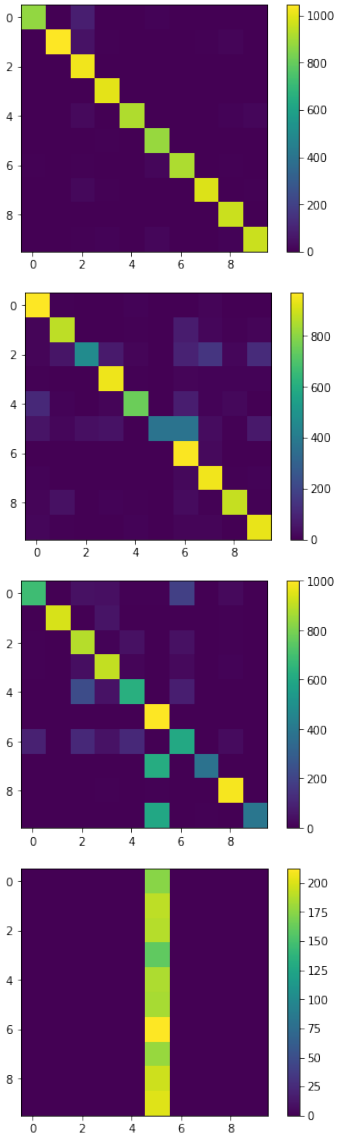

Figure 3. Confusion matrices of $F_{extract}(\cdot)$ on X_{val}. From top to bottom: MNIST, KMNIST, Fashion MNIST, notMNIST.

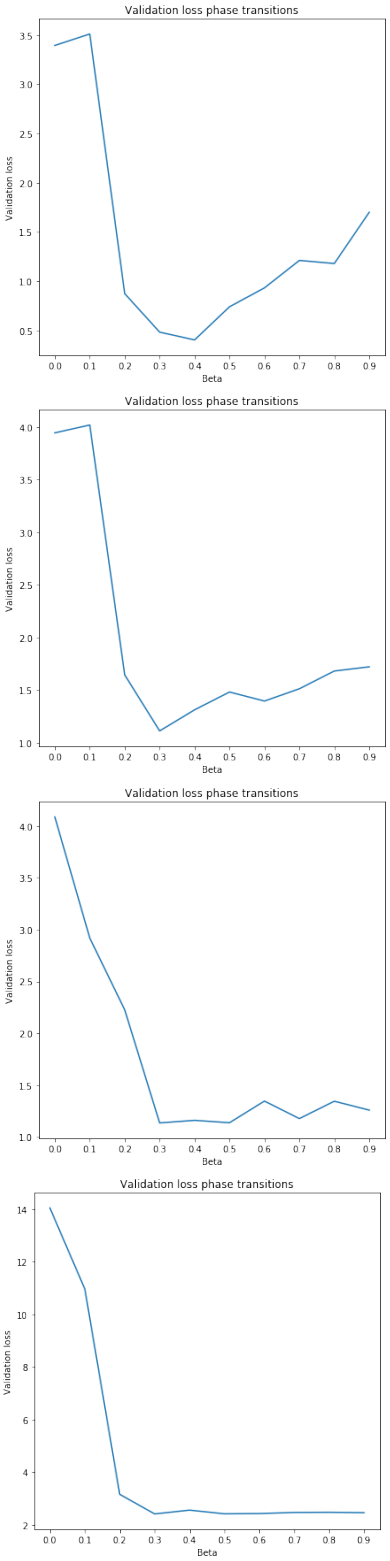

Figure 4. Loss phase transitions when β is varied. From top to bottom: MNIST, KMNIST, Fashion MNIST, notMNIST.

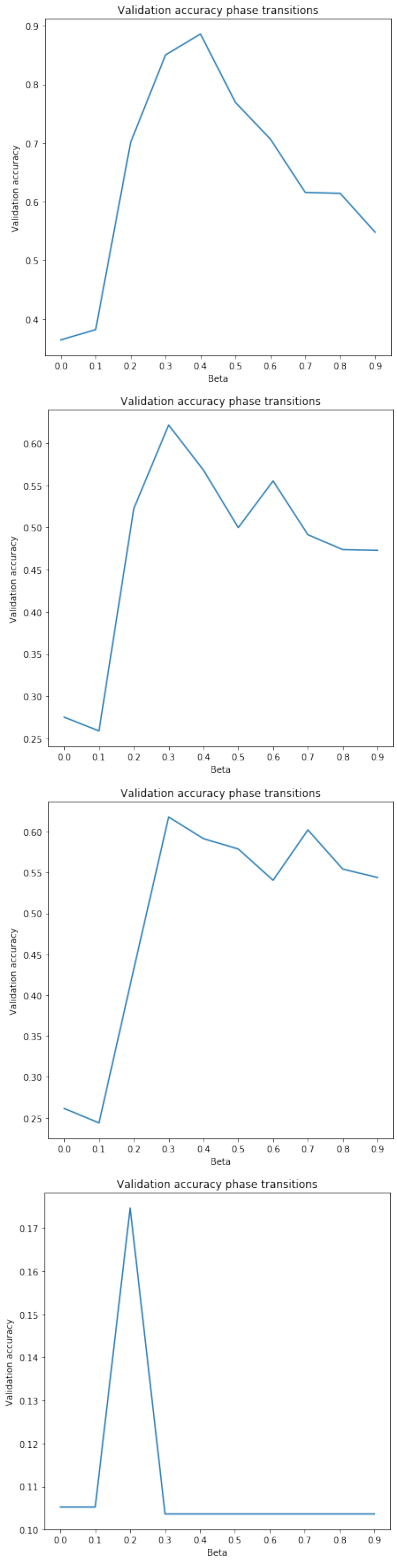

Figure 5. Accuracy phase transitions when β is varied. From top to bottom: MNIST, KMNIST, Fashion MNIST, notMNIST.

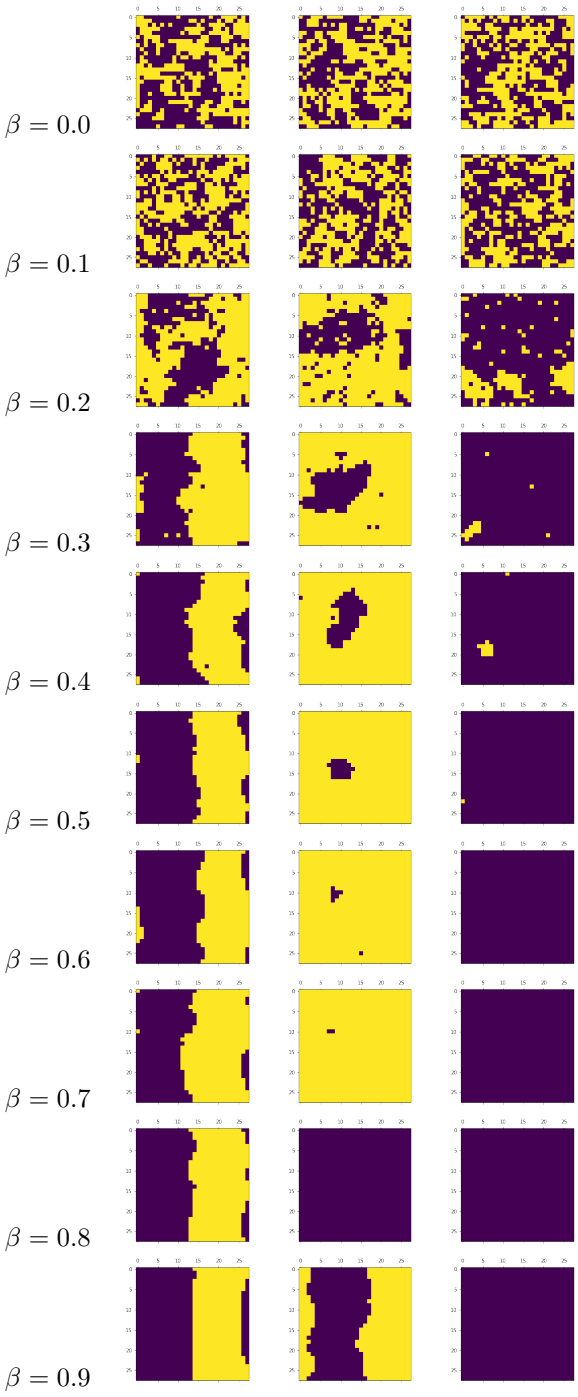

Figure 6. Examples of images from an Ising model simulation at various β parameters.