FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Štěpán Procházka

Adversarial Examples Generation for Deep Neural Networks

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis:	Mgr. Roman Neruda, CSc.
Study programme:	Computer Science
Study branch:	General Computer Science

Prague 2018

TODO Dedication.

Title: Adversarial Examples Generation
for Deep Neural Networks

Author: Štěpán Procházka

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Roman Neruda, CSc., Department of Theoretical Computer Science and Mathematical Logic

Abstract: TODO Abstract.

Keywords: machine learning adversarial examples evolutionary algorithms deep learning

Contents

Introduction

The effort to automate various processes in our lives has always been one of the key concepts of scientific research. The automation of mechanical work has already been solved to great extent by advances in engineering. On the contrary, automated processing of information has been solved just partially. Well defined tasks i.e., tasks with fully defined behaviour can be solved and the challenge lies just in effectivity of the solution. On the other hand, models solving tasks based on raw real word data, human level input and output or some degree of fuzziness are yet to be found or, if they exist, suffer from several shortcomings. One of those shortcomings is vulnerability to adversarial examples i.e., artificially created inputs misinterpreted by those models. In this thesis, we will cover the task of generating adversarial examples for the models used for classification in computer vision.

TODO 800 chars artificial intelligence -¿ machine learning -¿ sota deep learning

TODO 800 computer vision and classification tasks

1. Underlying Theory

In this chapter we will present theoretical background of various subjects relevant to this writing. We will give a compact overview of the task of image classification in computer vision with concrete examples of available datasets. We will cover various artificial intelligence paradigms with emphasis on deep learning for image classification and image generation, and evolutionary algorithms as a state space search method. Adversarial example generation methods will be discussed in the following, separate chapter for their importance with respect to this work.

TODO Overview - AI, ML, Optimization, Search - context and structure

- Artificial Intelligence (take out tasks, AI as methods) : Tasks — Classification - Regression - Sampling unknown distribution - Planning - Translation : Methods - Graph Search - Random/Beam/Local Search - Genetic Algorithms — Evolutionary Algorithms

- SVM : Machine Learning - Neural Networks — Deep Learning - Recurrent Neural Networks - Adversarial Examples

1.1 Classification Tasks in Computer Vision

The field of computer vision, sometimes percieved as a part of the artificial intelligence, examines the machine processing of imaging data. The basics of the field were laid down in the 1960s, the golden years of artificial intelligence, with ambitious goal to build systems with universal understanding of visual concepts. A multitude of subtasks of this ultimate goal were solved to a large extent; however the task as a whole remains to be solved at the time of writing. The computer vision is experiencing boom once again with the advent of deep learning and GPU accelerated computation.

TODO early works in CV - preset convolution filters, HOG Early works in computer vision were mostly based on deterministic algorithms. (TODO 2D convolution, fourier transformation for denoising (band pass filters), HOG for detection, matching etc, eigenfaces)

Based on the previous approaches in CV, such as the use of convolutional filters, in conjunction with increasing computational power and advancements in machine learning, such as reinvention of gradient backpropagation, the convolutional neural networks emerged (see 1.2). Those models outperformed former state of the art methods i.e., SVMs with RBF kernels or SIFT with Fisher vectors, in terms of accurracy (Krizhevsky et al. [2012]) and enabled wider range of tasks to be solved. The tasks that are being solved by CNNs in CV are image classification, object detection, semantic segmentation, image generation, style transfer, image captioning, etc. We will elaborate more on image classification tasks, available datasets and respective state of the art results.

TODO list of symbols and definitions

1.1.1 Task Definition

The task of image classification lies in assigning one and only one label l from the set of possible output labels L to each input image I from the space of all possible

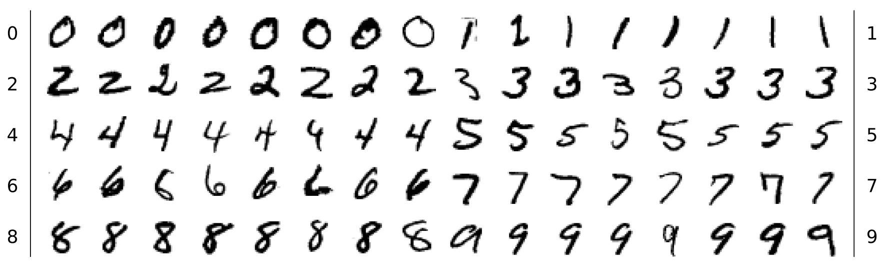

Figure 1.1: MNIST samples

images **I** of which we often think as a unit hypercube i.e., $[0, 1]^{h \cdot w \cdot c}$ with h, w, c being input images height, width and number of channels (depth) respectively. Task with input varying in shape can be transformed to the canonical classification task by adding preprocessing step which unifies shape e.g., reshaping using bilinear interpolation or conversion to common colour space, or by building a multitude of solutions, each targeted at specific shape of input data.

1.1.2 MNIST like tasks

The whole family of datasets for benchmarking of image classification solutions has been created. For the sake of interchangeability all of them share the same input size of $28 \times 28 \times 1$ (rectangular grayscale image) and often same dataset size and structure with 60,000 training and 10,000 testing examples. Those datasets, particularly the original MNIST itself, became widely used benchmarking datasets for they are rather small in terms of the number of examples and their shape, enabling fast prototyping and experimentation.

MNIST The dataset comprises of handwritten digits – a selected subset of the National Institute of Standards and Technology [2010] database. The database consists of approximately 800,000 handwritten alphanumeric characters written by 3600 writers, namely high school students and the United States Census Bureau employees. The subset has been chosen so that the amount of samples written by students compared to the Census Bureau employees is equal in each dataset split. Moreover the sets of writers chosen for training samples and testing samples are disjoint. The chosen samples are normalized with following steps – each character is reshaped to fit $20 \times 20px$ patch preserving the aspect ratio and the center of mass of this patch is aligned with the center of larger $28 \times 28px$ patch creating final sample. The classification accuracy of over 99% was reached by Lecun et al. [1998] and their LeNet model. Due to increasing capabilities of machine learning approaches, MNIST became obsolete and is no longer a valid benchmarking dataset for performance comparison nowadays. 1.1

EMNIST The Extended MNIST (EMNIST) dataset contains alphanumeric characters and consequently is seen as an extension of the MNIST dataset. It comes from the same database and was created by mimicking the steps used for MNIST sample preprocessing. However, the procedure differs sligtly and consequently MNIST is not a subset of EMNIST. The dataset comes in several layouts. The EMNIST Complete consists of approximately $700,000$ training samples and

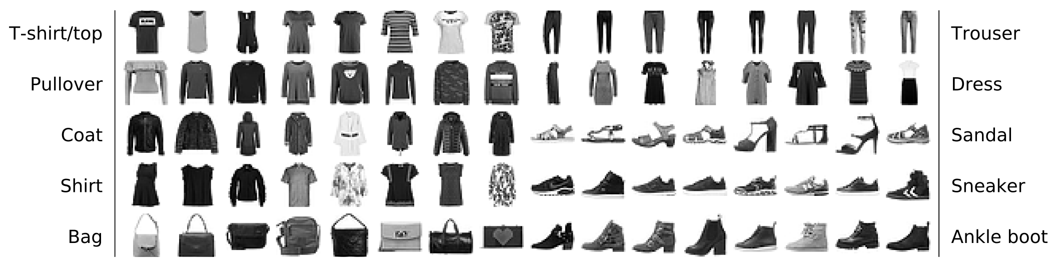

Figure 1.2: Fashion MNIST samples

115, 000 testing samples each classified to one of 62 classes (10 digits, 26 lower-case and 26 upper-case letters). The EMNIST Merge is a variation of the complete EMNIST with visually ambigous letter classes e.g., *C, K, P, S* merged, leading to 47 classes in total. The EMNIST Balanced comprises of 112, 800 training and 18, 800 testing samples in 47 classes (using merging), with equal sample frequency across classes. Appart from those datasets, there are EMNIST Digits, EMNIST Letters and EMNIST MNIST, dataset sharing the layout of original MNIST dataset (Cohen et al. [2017]). Despite being substantialy harder task than MNIST, EMNIST is not widely used.

Fashion MNIST The Fashion MNIST shares the layout of MNIST dataset, having the same number of same shaped train and test examples as well as the same number of classes. The dataset comprises of grayscale thumbnails of pieces of clothes in ten classes *T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot* 1.2. This dataset seems to be promising replacement of the original MNIST as it is defining harder and more computer vision relevant task (images of real world objects rather than manmade symbols), while preserving moderate hardware requirements and model complexity demands. Xiao et al. [2017a]

1.1.3 Photo datasets

Due to the need to classify real world imagery i.e., data with substantial amount of noise, lacking quality or having non-trivial composition such as multiple objects in the scene, varying viewing angles or conditions, or heterogenous background), the datasets of real world photographs were created. Those datasets range from collections of thousands of low resolution thumbnails in dozens of classes to millions of reasonably large pictures in hierarchical systems of classes.

CIFAR The CIFAR dataset comes in two versions, namely CIFAR-10 and CIFAR-100, named after number of classes present. Both datasets share the same example shape i.e., 32 × 32 (rectangular color image) and dataset layout comprising of 50,000 training and 10,000 testing images with balanced class occurence. CIFAR-10 consists of following classes – *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck* 1.3, while CIFAR-100 introduces hierarchical system of 20 superclasses e.g., *fish, small mammals, tress, large carnivores* each subdivided into 5 classes e.g., *fish – aquarium fish, flatfish, ray, shark, trout* (see Krizhevsky [2009].

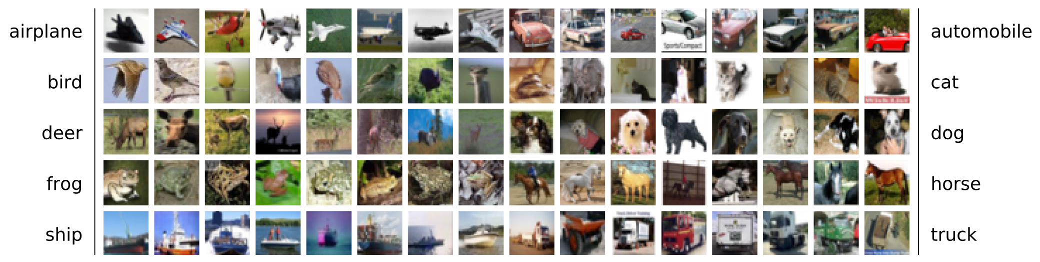

Figure 1.3: CIFAR 10 samples

ImageNet TODO - few images - results - ILSRVC

1.2 Deep Learning in Image Classification

The deep learning is a branch of machine learning characterized by utilization of multilayered neural networks. Due to the complexity of deep networks and solutions tailored to solve tasks with fuzzy real world data, deep learning approaches are often not backed up by strong methematical theory; however they are proving successful in solving those tasks and provide current state of the art solutions in the field of computer vision, speech recognition and synthesis, machine translation, etc. Nowadays the goal of deep learning research lies in explanation of inner workings of deep models, which may result in constructing more complex and more capable machine learning solutions. See Goodfellow et al. [2016] for in depth introduction on deep learning.

1.2.1 Machine Learning

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. Mitchell [1997]

Machine learning is an Artificial Intelligence paradigm based on the concept of knoledge extraction from observed states or experience, followed by application of obtained knowledge to new observations. More formally the machine learning solution, often reffered to as *model* M, can be be percieved as the approximation of probability distibution P, optionaly conditional, of random variable $\mathtt{x}$, representing the task being solved. The model is often implemented by a parametric differentiable composite function M_θ and the process of finding the solution for a given task is called *training* and lies in finding the optimal set of parameters θ. The process of training relies on training data i.e., set of observed states, samples of random variable $\mathtt{x}$. The optimal set of parameters is searched for using maximum likelihood estimate principle i.e., the estimate of likelihood that the probability distribution of model P_{M_θ} matches the probability distribution of random variable $\mathtt{x}$. This process is implemented as a minimization of loss function, often using gradient descent algorithm. Given the characteristics of the training data, machine learning tasks can be split into the following two categories.

supervised training data takes form of (x, y) i.e., input – output or querry – answer example pairs. The goal is to approximate the conditional probability distribution $P(y|x)$ (*classification, encoding*); in case of *reinforcement learning*, the dataset itself is not available, and the ground truth comes in form of full description of the dynamic environment in which the model operates

unsupervised training data takes form of singletons x i.e., samples of unknown distribution and the the goal is usually to find unknown structure (*clustering*), be able to *generate* new samples i.e., approximate the probability distribution $P(x)$, or transform the data to some latent space with usefull properties (*compression, encoding, feature extraction*)

1.2.2 Loss Functions

Given the probability distribution $\bar{P}$ of the training data – samples of unknown distribution P, and probability distribution $\bar{P}_{M_\theta}$, the loss function quantifies the likelihood of $\bar{P}_{M_\theta}$ matching the distribution P, using the estimation $\bar{P}$. Frequently used loss functions are Kullback-Leibler divergence, often refered to as cross-entropy, used for the purpose of classification tasks; mean-squared error for regression tasks, or more advanced loss functions such as conditional random fields (*CRF*) loss or connectionist temporal classification (*CTC*) loss. Only the two former will be discussed in detail for their relevance to this writing.

Kullback-Leibler divergence For the discrete probability distributions, the *KL-divergence* has the following form.

$$D_{KL}(\bar{P} \parallel \bar{P}_{M_\theta}) = -\sum_i \log \bar{P}(i) \frac{\bar{P}_{M_\theta}}{\bar{P}(i)} \tag{1.1}$$

It holds that *KL-divergence* is always non-negative (TODO reference Gibbs inequality) and equal to zero if and only if $\bar{P} = \bar{P}_{M_\theta}$. It is worth to be noted that *KL-divergence* is not symmetric, hence it is not a distance measure.

Mean squared error

$$MSE(\bar{P}, \bar{P}_{M_\theta}) = \frac{1}{n} \sum_{i=1}^{n} (\bar{P}(i) - \bar{P}_{M_\theta})^2 \tag{1.2}$$

It holds that *MSE* is always non-negative and equal to zero if and only if $\bar{P} = \bar{P}_{M_\theta}$.

1.2.3 Optimizers

The optimizer controls the process of finding optimum of the loss function. Unlike classical methods of mathematical optimization working with fully defined functions (or probability distributions they generate), optimizers in machine learning work with finite set of samples of unknown distribution. As a result, finding the global optimum on the training set is not the preferable goal, as it brings the issue of overfitting – relying on characteristics specific to the training data

$\bar{P}$ which are not representative in context of the whole unknown distribution P. To fight overfitting, methods of *regularization* are often employed. Due to the hardware limitations, it is often not possible for the optimizer to work with the whole training set. As a result, sampling of dataset is employed in the form of batching, and the process of training is performed in *epochs* – iterations over the whole training set in batches. Optimizers frequently used during training of deep learning models are stochastic gradient descent (optionaly with momentum or Nesterov momentum), RMSProp, AdaGrad, Adam, etc.

Gradient Descent The gradient descent is a state space search method. It navigates the space of the parameters of the machine learning model by iteratively taking steps in the opposite direction of the gradient of the loss function, hence minimizing it. The step size is driven by *learning rate* parameter. The algorithm does not guarantee finding global optimum. When using batching, we talk about stochastic gradient descent for we have access only to the approximation of the loss function in each step.

Algorithm 1 Stochastic gradient descent (optionaly with Neterov momentum)

```
$\alpha \leftarrow$ learning rate
$[\beta \leftarrow momentum, v \leftarrow 0]$
repeat
    Sample a minibatch of $m$ training examples $(x^{(i)}, y^{(i)})$
    $[\theta \leftarrow \theta + \beta v]$
    $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)};\theta), y^{(i)})$
    $[v \leftarrow \beta v - \alpha g]$
    $\theta \leftarrow \theta - \alpha g$
until early stopping criterion is met
```

Commands in brackets implements usage of Nesterov momentum with SGD. Each step, we first change parameters according to the accumulated momentum, than we estimate gradient of the loss function as in basic SGD, update momentum and perform step according to estimated gradient.

Adam The *Adam* optimizer is one of optimizers derived from basic SGD Kingma and Ba [2014]. It addresses the shortcomings of the stochastic gradient descent, mainly high variance in loss function gradient estimation and inability to effectively navigate certain landscapes. Adam keeps track of the first and the second moment estimates of the loss function gradient, using exponential moving average.

TODO vectors bold

Adam estimates the mean of gradient in the same way as SGD with momentum. Moreover, the second moment estimate provides adaptive learning rate for each parameter in θ resulting in faster convergence during training and more stable behaviour.

Algorithm 2 Adam optimizer

```
$\alpha \leftarrow$ learning rate
$\beta_1 \leftarrow$ mean decay rate, $\beta_2 \leftarrow$ variance decay rate
$s \leftarrow 0, r \leftarrow 0, t \leftarrow 0$
**repeat**
    $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^{(i)};\theta), y^{(i)})$
    $t \leftarrow t + 1$
    $s \leftarrow \beta_1 s + (1 - \beta_1)g$
    $r \leftarrow \beta_2 r + (1 - \beta_2)g^2$
    $\hat{s} \leftarrow s/(1 - \beta_1^t)$
    $\hat{r} \leftarrow r/(1 - \beta_2^t)$
    $\theta \leftarrow \theta - \frac{\alpha}{\sqrt{\hat{r}+\epsilon}}\hat{s}$
**until** early stopping criterion is met
```

1.2.4 Regularization

As stated before, the issue of overfitting is adressed by means of regularizations – methods, that improve model generalization ability (performance on unseen data), while not neccessarily changing the performance on training data. Frequently used algorithms are:

early stopping policy terminating training process if model performance on validation set visibly stalls or worsens

p-norm regularization additional term in loss function computing $p - norm$ of model parameters, forcing them to maintain mean close to zero and low variance; $l2$ and $l1$ norms are often used

dataset augmentation careful application of transformations to training data provides mean of enlarging training set, and consequently improving model performance; shifting, flipping, random cropping, addition of random noise are broadly used augmentation methods, they need to be tailored to specific task to preserve label of augmented example

ensembling training multiple models with different initialization, different training process or even of different architectures and employing a voting scheme for those models often leads to better results; however this approach comes with significant computational power demands

dropout TODO

1.2.5 Layers

Neural networks are often thought of as layered structures with layers being transormations of tensors. The deep neural networks are composed of tenths or even hundreds of those layers. Various types of layers serve for feature extraction, non-linear transformation, normalization, reshaping, etc.

TODO list of symbols ???

Dense The most basic and arguably the first layer invented is a dense layer. Given the input tensor

Activation functions

Convolution

Pooling

Normalization

Shortcuts

1.2.6 Known Architectures

- alexnet - resnet - densenet

1.3 Evolutionary Algorithms as Space Search Algorithm

- state space search - biologicaly inspired - basic terminology

2. State of The Art Approaches

TODO rename

2.1 Adversarial Examples for Deep Learning Models

- noise + input - other than image - known methods
TODO topology - black/white box, iterative/single-shot, targeted/non-targeted

2.2 Gradient based methods

2.3 Variational Auto-Encoders

3. Our Solution

3.1 Pure Evolutionary Algorithms

3.2 Hybrid Methods

4. Experiments

In order to empirically assess the performance of proposed solutions we have carried out a multitude of experiments on the Fashion MNIST dataset. This particular dataset was chosen for it is simple enough to enable thorough examination of various scenarios using k-fold cross-validation and measurements of results on statistically significant number of examples (often whole Fashion MNIST test set), yet more complex then MNIST and the like (1.1.2). Following the results of Fashion MNIST benchmark (Xiao et al. [2017b]) we have chosen two high scoring model architectures of varying nature – namely:

SimpleNet Simple wide shallow network with three convolutional layers of depth 32, 64 and 128 respectively and one hidden dense layer of size 128. Each convolutional layer uses square kernel of size 3 and stride 1 with *same* padding and is followed by batch normalization and ReLU activation. Max pooling with kernel of size 2 and stride 2 is added in-between each pair of convolutions for subsampling. Flattened output of last convolutional layer is followed by ReLU activated hidden dense layer and final output layer of size corresponding to the number of classes. Dropout with rate 0.3 is added after each max pooling and in front of hidden and output dense layers. Model is trained with Adam optimizer (Kingma and Ba [2014]) with default (recommended) parameters for 128 epochs. Initial learning rate 0.001 is divided by 5 in 50% and 75% of training process. To the best of our knowledge the name *SimpleNet* does not refer to any well-known model architecture and was chosen for easier distinction of used architectures when refering to them.

DenseNet Specifically *DenseNet-BC* of depth 52 with growth rate $k = 8$ and 0.5 compression factor, representing deep narrow convolutional network. Dropout of rate 0.2 is used as proposed by authors as well as recommended training parameters i.e., SGD with Nesterov momentum of 0.9 and learning rate of 0.1 divided by 10 in 50% and 75% of training process which takes, again, 128 epochs. See Huang et al. [2016]. 4.1

Both models use weight decay of 0.0001 as additional regularization method as well as standard input preprocessing - mean subtraction and standard deviation division with channel-wise precomputed moments on training data (TODO citation). Batch size of 128 is used during training and no early stopping method is employed. 4.2

Training of both architectures is carried out using stratified 10-fold cross-validation resulting in 20 target models trained with varying random initialization on differing training sets.

TODO two plots - box-plot of test accuracies and losses - example training validation loss and accuracy

Experiments overview In the following sections we are going to examine following approaches towards generating adversarial examples for image classifiers.

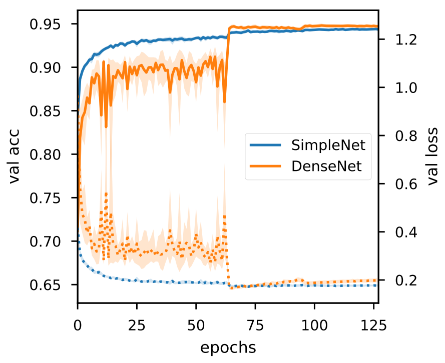

Figure 4.1: Training progress

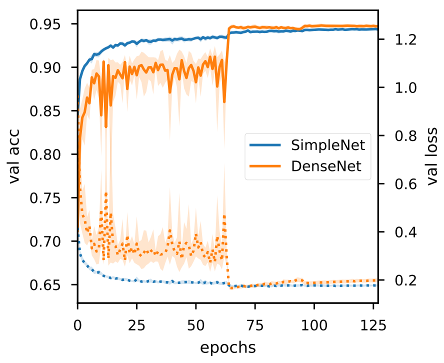

Figure 4.2: Another figure

Firstly, white-box fast gradient sign method attack serving as both implementation sanity check and baseline is carried out. Next the transferability of adversarial examples is examined using surrogate models. Those approaches include transfering adversarial examples between models trained on the same dataset – the most naive approach, training surrogate model on outputs of target model and finally training focused surrogate model as a binary classifier for specific classification category in one-vs-all manner. For each approach based on use of surrogate models, the transferability of adversarial examples between models of the same architecture as well as models of differing architectures is assessed. Appart from pure gradient methods, evolutionary algorithms are employed as a state space search method, searching for adversarial examples for target model. Finally using both gradient based methods and heuristic state space search together, the performance of hybrid methods is measured. For some experiments one *SimpleNet* and one *DenseNet* model is chosen, taking median models with respect to accuracy, we will refer to them as *median Simple/DenseNet*. In the same manner a single class is chosen, median in terms of model prediction accuracy and attack performance, for use in some experiments.

Each approach is tested under combination of following conditions - either non-targeted and targeted attack to each class is carried out, while at the same time generating single or universal adversarial pattern is tested.

4.1 Fast Gradient Sign Method Approaches

Iterative FGSM is applied to each example in Fashion MNIST test set. FGSM with step size of $\frac{1}{255}$ is used, which corresponds to shift by one shade of grey in resulting adversarial example. Maximal allowed pixel-wise difference is set to 0.1, which corresponds to approximately 26 shades of gray difference in either direction. The following experiments compare adversarial attack performance for various model architectures, their respective training set and characteristics of the attack itself.

4.1.1 White-Box Attack

The white-box iterative FGSM is run against the target models directly. Following box plots show the distribution of the number of steps (*y-axis*) needed to

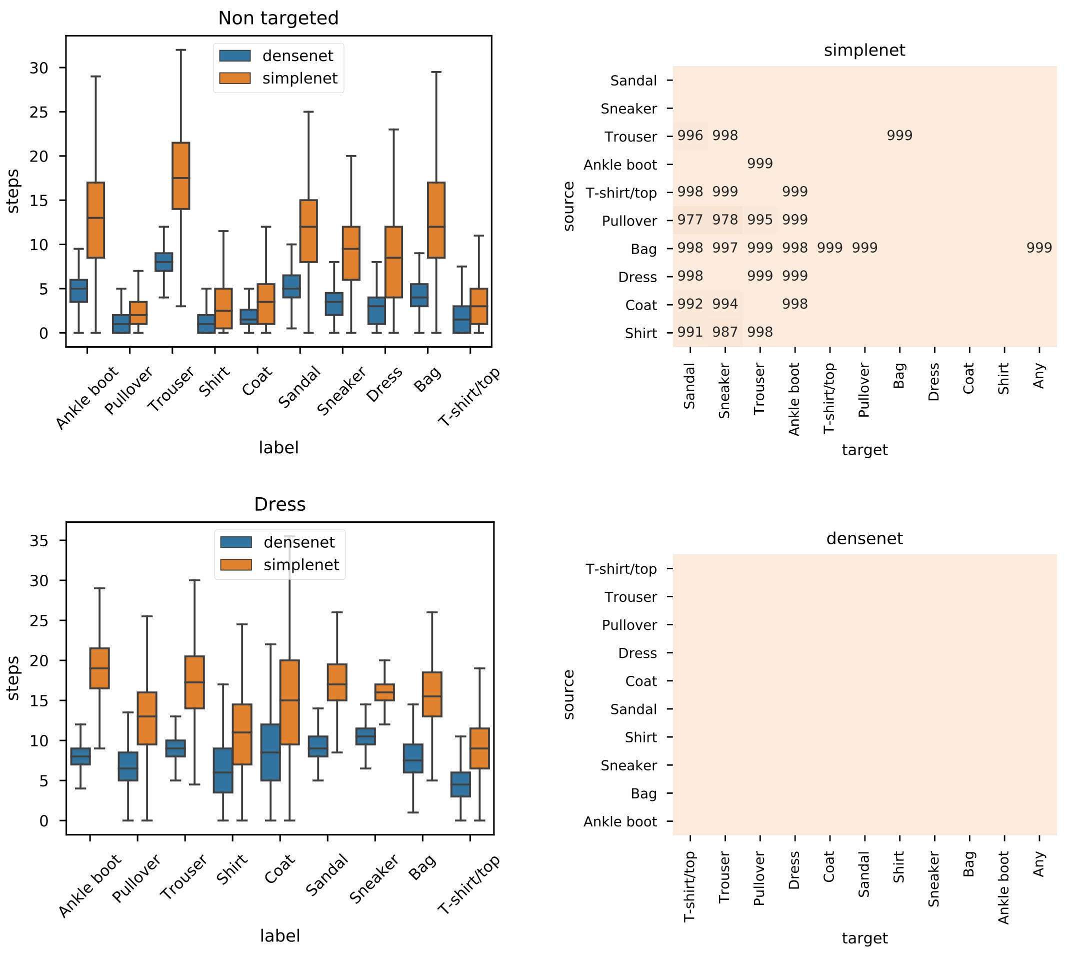

Figure 4.3: FGSM White-box

be taken to obtain adversarial example from source label (*x-axis*) to target (*plot title*), those results are collected on the whole Fashion MNIST test set using cross validated target models i.e., for each class – architecture pair, the median step count for each test set example is taken, resulting in 1000 datapoints. Only valid datapoints i.e., those representing successfully generated adversarial examples are used to render each box. Presented heatmaps than show success rate of attack for each source – target label pair for given architecture (*plot title*). Blank cell in heatmap represents maximal possible value i.e., 1000.

Singleton Following from presented results 4.3 we can see that white box attack against single example proves to be successful in both targeted and non-targeted scenarios for each examined architecture. *Simplenet* architecture proves to be slightly more resilient both in terms of success rate and iterative FGSM step count. TODO comparison with confusion matrix

TODO link confussion matrix with boxplots - rows, whole, columns whatever TODO maintain ordering of confussion matrices across experiments

Universal TODO studijni obor u statnic

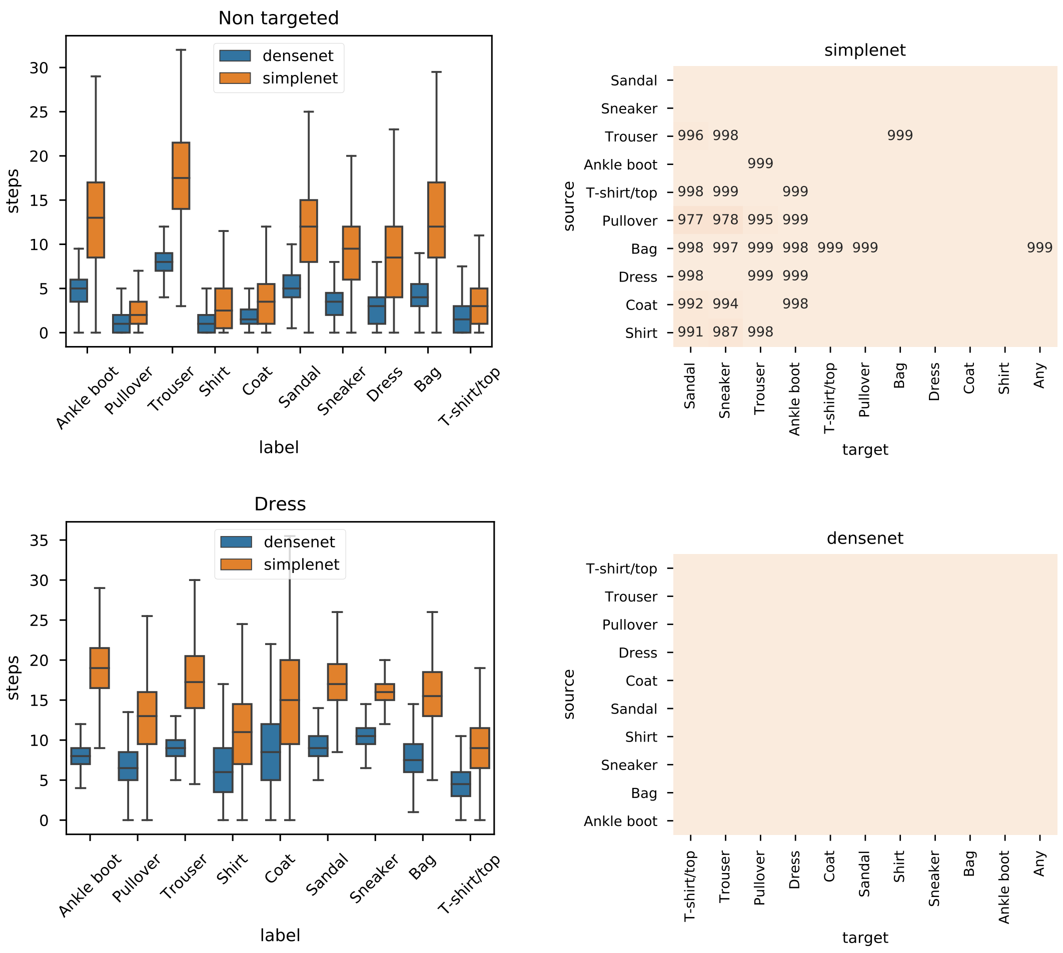

Figure 4.4: TODO

4.2 Surrogate attack

Following results are reflecting the performance of transferability of adversarial examples using various types of surrogate models.

TODO success rate comparison (?? mean success rate of method or per architecture??)

Experiments will be carried out for those scenarios:

- FGSM (white-box, target model)
- FGSM (surrogate, different seed model)
- FGSM (surrogate, teacher-student model)
- FGSM (surrogate, specific binary teacher-student model)
- EA (black-box, pure EA)
- hybrid (surrogate, EA + pre-trained surrogate)
- hybrid (surrogate, EA + on-demand-trained surrogate)

Each combination of following options will be tested:

- targeted vs non-targeted
- single image vs multi-image vs generalization (unseen images)

Each experiment will be cross-validated using 10-fold cross-validation.

Architectures - Simple CNN - 3 cnn layers with max-pooling and one hidden dense layer - DenseNet-BC-8-52

Datasets - Fashion MNIST - Cifar-10 - bude-li čas

median simple_cnn model -¿ fold_7 median dense_net model -¿ fold_0

experiments on test set - unseen data - common scenario in real world FGSM white-box, target model - confidence bound 0.5, 0.95 (for surrogate) - whole test set, targeted/non-targeted, single-image - multi-image - whole class - outcomes (metrics) ?? - median/mean/std/quantiles of step count for iterative FGSM ? - median/mean/std/quantiles of added noise? (MSE?) - rate of success (will be close to 1) - time ? - some examples? which ones (unsuccessfull ones if any?) - show visual FGSM surrogate (different seed) - cross-validation folds cross comparison (whole/partial?) - 2 vs 20 (wiht 0.95 bound) - median model - median of accuracy - multi-image - whole class FGSM surrogate (teacher-student) - 2 median modely * 10 fold * 2 architectures - whole dataset (resplitted randomly) - 1/10th of dataset (resplitted randomly) - multi-image - whole class FGSM surrogate (binary teacher-student) - 2 median-model * 10 fold * 2 architectures * 2 big small data (just one median class) - (maybe if really bad - balance dataset) EA (black-box, pure EA) - 10 random (really random) samples from test set - multi-image - vs median models hybrid - pre-trained surrogate - 2 median-surrogates * 2 binary/full * 10 random sample * ... hybrid - on-demand surrogate - 2 binary/full * 10 random sample * 2 * policies

4.3 Fast Gradient Sign Method

White-box attack

4.4 Evolutionary Algorithm

4.5 Hybrid methods

Conclusion

Bibliography

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL `http://arxiv.org/abs/1702.05373`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL `http://www.deeplearningbook.org`.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. URL `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`. Last visited 2018-07-02.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill Education, 1997. URL `http://www.cs.cmu.edu/~tom/mlbook.html`.

National Institute of Standards and Technology. Nist special database 19, 2010. URL `https://www.nist.gov/srd/nist-special-database-19`. Last visited 2018-07-02.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017a. URL `http://arxiv.org/abs/1708.07747`.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: Benchmark, 2017b. URL `https://github.com/zalandoresearch/fashion-mnist#benchmark`. Last visited 2018-07-03.

List of Figures

List of Tables

List of Abbreviations

A. Evgena Framework User Documentation

B. Attachments

B.1 First Attachment