Adversarial Learning

Building Robust Vision Models

Slide Deck and Code

http://bit.ly/adv_learn_tfug20

Session Agenda

- Introduction
- CNN Brief
- Transfer Learning Brief
- Adversarial Attacks
- Neural Structured Learning
- Hands-on Tutorial


Introduction

About Me

Dipanjan Sarkar

Data Science Lead, Author, Google Developer Expert - ML

APPLIED MATERIALS®

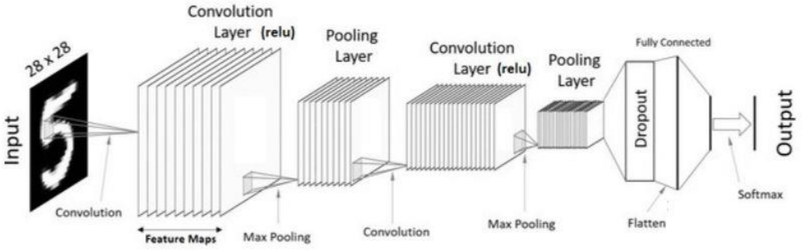
Springboard

Experts

Machine Learning


CNN Brief

CNN Architecture Components



- **CNNs have a stacked layered architecture of several convolution and pooling layers**
- **Convolution layer**
  - Consists of several filters or kernels
  - Passed over the entire image in patches and computes a dot product
  - Result is summed up into one number per operation (dot product)
- **Pooling layer**
  - Downsamples feature maps from conv layers
  - Typically max-pooling is used which selects the max-pixel value out of a patch of pixels
- **Activation layer**
  - Feature maps \ pooled outputs are sent through non-linear activations
  - Introduces non-linearity and helps train via. backpropagation

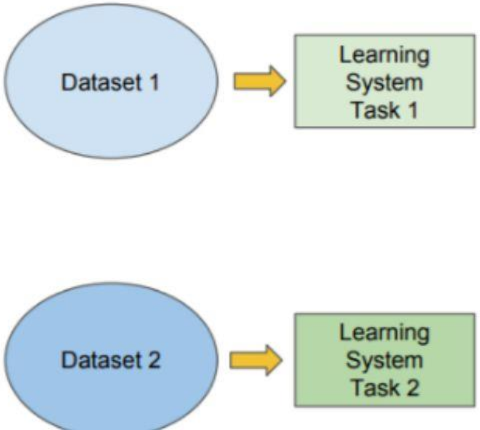

Transfer Learning Brief

Traditional ML vs. Transfer Learning

Traditional ML vs Transfer Learning

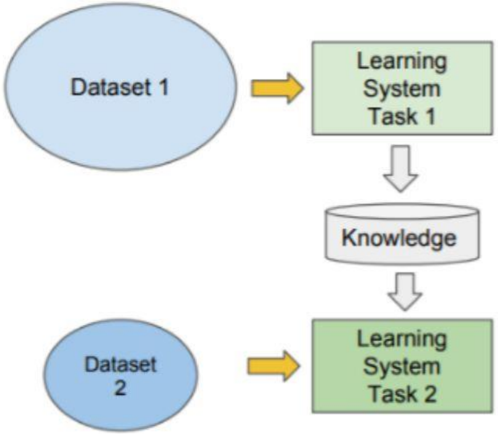
Traditional ML

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



The power of Transfer Learning

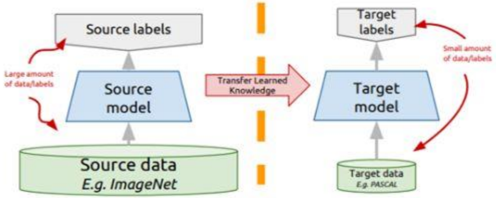
Transfer learning: idea

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

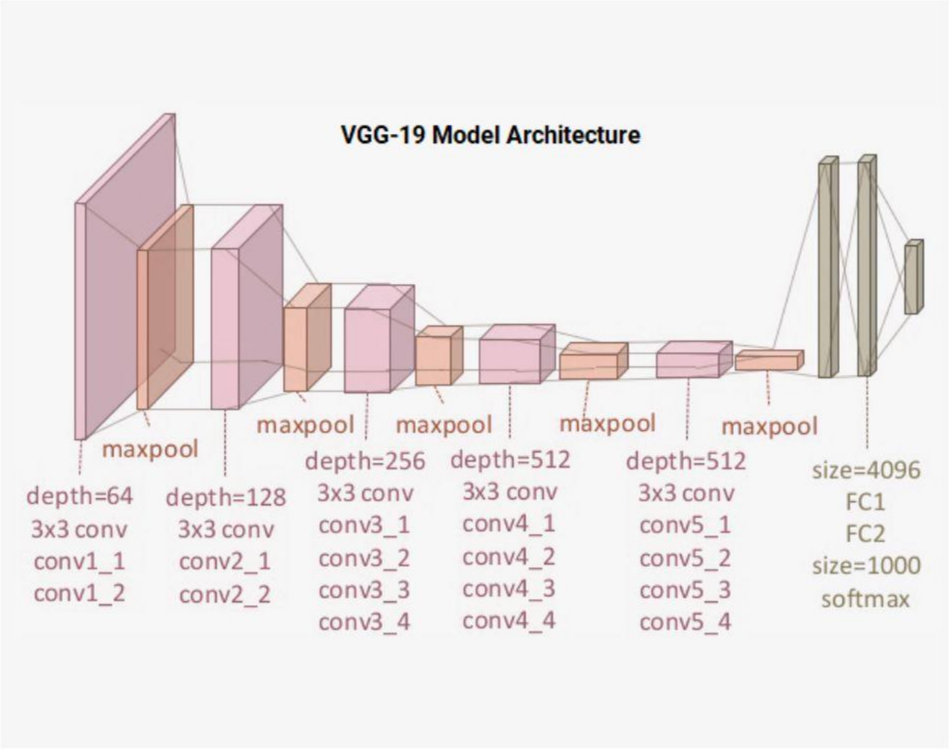
Variations:

- Same domain, different task
- Different domain, same task



- **Leverage a pre-trained deep learning model (which was trained on a large dataset — like ImageNet)**
- **Solve the problem of apparel classification in Fashion-MNIST by applying and transferring its knowledge in the context of our problem**
- **Two approaches**
  - Frozen pre-trained model as a feature extractor
  - **Fine-tuning pre-trained model on our data**

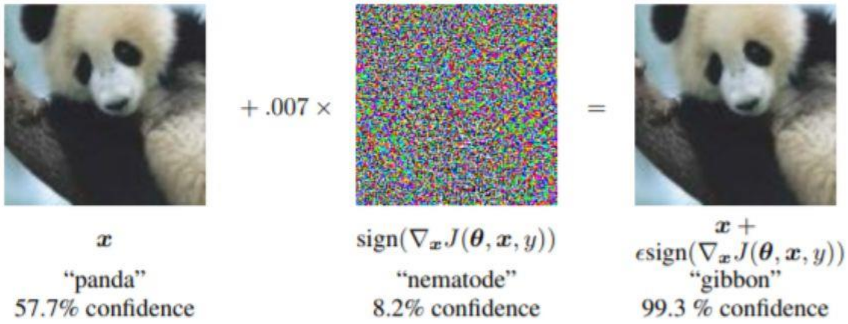
Pre-trained CNN - VGG-19



- VGG-19 model is a 19-layer (convolution and fully connected) deep learning network built on the ImageNet database
- 16 convolution layers using 3 x 3convolution filters along with max pooling layers
- 2 fully connected hidden layers of 4096 units in each layer followed by a dense layer of 1000 units
- We focus on intermediate layers for extracting the right representations

Adversarial Attacks

Adversarial Attacks



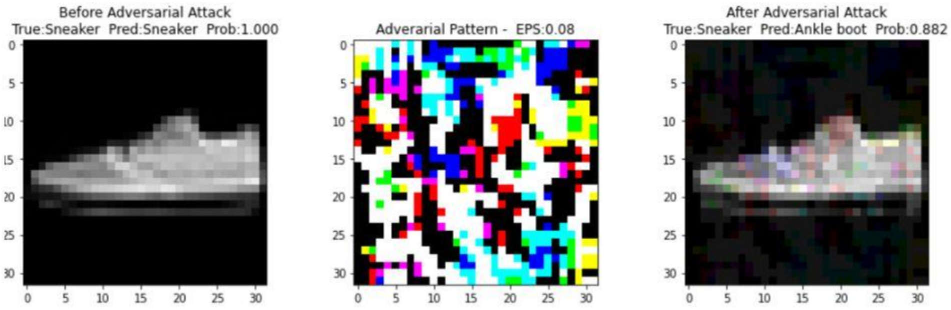
- **Purposely adding perturbations or noise in the data to fool the model**
- **Synthetically created on top of an input image**
- **Attacker adds small perturbations (distortions) to the original image**
- **These notorious perturbations are indistinguishable to the human eye, but causes the network to fail**

Adversarial Attacks - Fast Gradient Sign Method

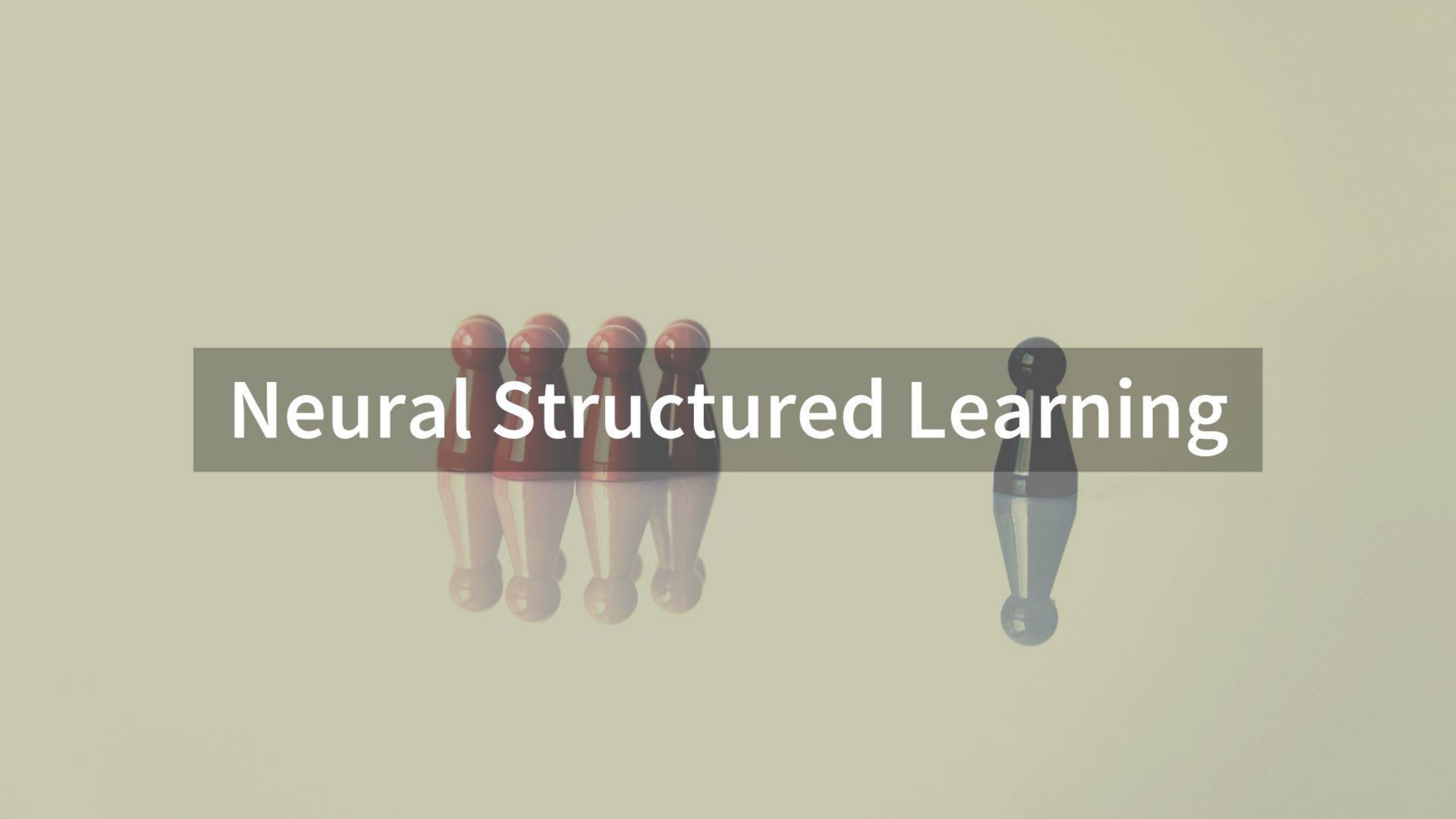
$$adv\_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$$

where

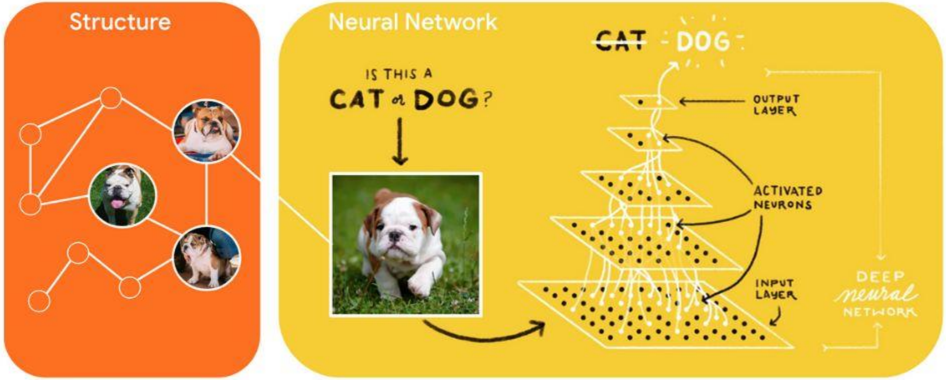
- adv_x : Adversarial image.
- x : Original input image.
- y : Original input label.
- ϵ : Multiplier to ensure the perturbations are small.
- θ : Model parameters.
- J : Loss.



- Uses the gradients of the neural network to create an adversarial example
- Objective is to create an image that maximizes the loss.
- Gradients of the loss with respect to the input image are taken
- A small multiplier (epsilon) is added to the sign of the gradients and added to the original image
- Goal is to fool an already trained model.

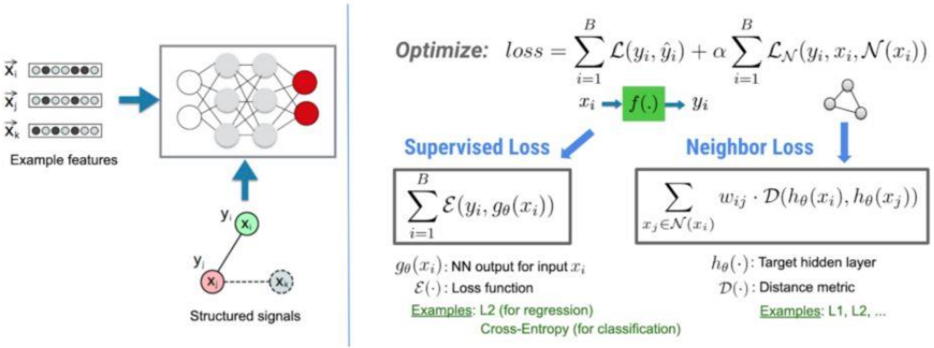

Neural Structured Learning

Neural Structured Learning



- A new learning paradigm to train neural networks by leveraging structured signals in addition to feature inputs
- Structure can be explicit as represented by a graph or implicit
  - Implicit structure can be created by leveraging nearest neighbors similar to input
  - Adversarial examples created by perturbations on inputs can also be used
- Structured signals are commonly used to represent relations or similarity among samples
- Models trained with adversarial perturbation samples have been shown to be robust against malicious attacks

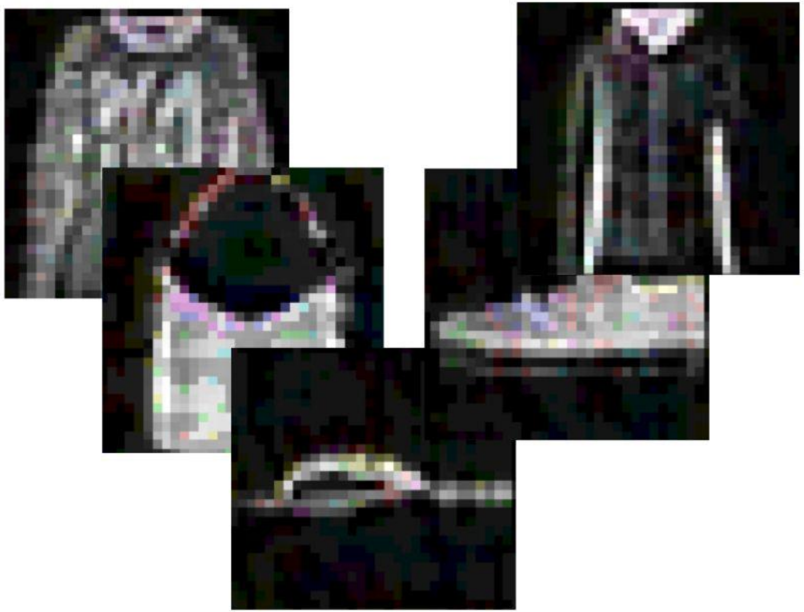
Neural Structured Learning - Methodology



- Structured signals e.g. generated adversarial examples, are used to regularize the training of a neural network
- Objective is to minimize total loss

  total_loss = supervised_loss + neighbor_loss
- Minimize supervised loss for accurate predictions
- Minimize neighbor loss to maintain the similarity among inputs from the same structure

Adversarial Learning with NSL



- Create implicit structures by generating adversarial examples
- Perform Adversarial Regularization

  total_loss = supervised_loss + adversarial_loss
- Minimize supervised loss for accurate predictions
- Minimize adversarial loss to maintain the similarity among inputs and their adversarial examples

Adversarial Learning with TensorFlow - NSL

```python
import neural_structured_learning as nsl

# Prepare data.
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Create a base model -- sequential, functional, or subclass.
model = tf.keras.Sequential(...)

# Wrap the model with adversarial regularization.
adv_config = nsl.configs.make_adv_reg_config(multiplier=0.2, adv_step_size=0.05)
adv_model = nsl.keras.AdversarialRegularization(model, adv_config)


# Compile, train, and evaluate.
adv_model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
adv_model.fit({'feature': x_train, 'label': y_train}, epochs=5)
adv_model.evaluate({'feature': x_test, 'label': y_test})
```

Read Data

Keras Model

Config Adv model

Compile

Fit

Eval

Adversarial Learning Hands-on Tutorial

References

- **Visuals & Content**
  - https://www.tensorflow.org/neural_structured_learning
  - https://medium.com/tensorflow/introducing-neural-structured-learning-in-tensorflow-5a802efd7afd
  - https://github.com/sayakpaul/Image-Adversaries-101
  - https://github.com/dipanjanS/convolutional_neural_networks_essentials
  - https://www.tensorflow.org/tutorials/generative/adversarial_fgsm
  - https://www.tensorflow.org/neural_structured_learning/tutorials/adversarial_keras_cnn_mnist
- **Research Papers**
  - https://github.com/dipanjanS/adversarial_learning_tfug2020/tree/master/papers

Stay in Touch!

LinkedIn

https://www.linkedin.com/in/dipanzan

GitHub

https://github.com/dipanjanS

Medium

https://medium.com/@dipanzan.sarkar