

# Adversarial Learning

Building Robust Vision Models

## Slide Deck and Code

[http://bit.ly/adv\\_learn\\_tfug20](http://bit.ly/adv_learn_tfug20)

# Session Agenda



Introduction



CNN Brief



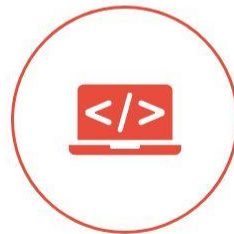
Transfer Learning Brief



Adversarial Attacks



Neural Structured Learning



Hands-on Tutorial

# Introduction

A person stands on the crest of a dark, rolling dune under a vast, golden sky at sunset. The sun is a bright, glowing orb in the upper center, casting a warm, orange light across the horizon. The landscape is composed of several layers of rolling dunes, with the foreground being the darkest. In the distance, a few silhouetted trees are visible on the horizon line. A semi-transparent grey rectangle is centered over the image, containing the word "Introduction" in white, bold, sans-serif font.



About Me

# Dipanjan Sarkar

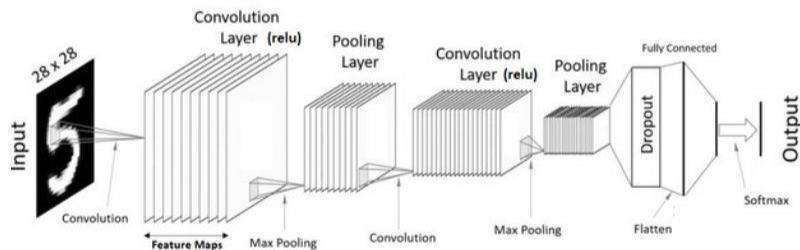
Data Science Lead, Author, Google Developer Expert - ML



An aerial, top-down view of a large, circular building. The building's floor is a light tan color, and it is surrounded by a dark blue-grey exterior wall. Numerous rectangular windows are arranged in concentric circles, creating a radial pattern. The windows in the inner circles are smaller, while those in the outer circles are larger. The central area of the building is a large, empty circle. In the center of this circle, the text "CNN Brief" is displayed in white, bold, sans-serif font, enclosed within a semi-transparent grey rectangular box.

# 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





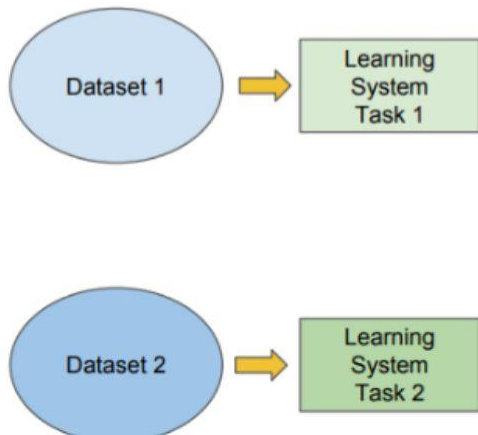
# Transfer Learning Brief



# Traditional ML vs. Transfer Learning

## Traditional ML

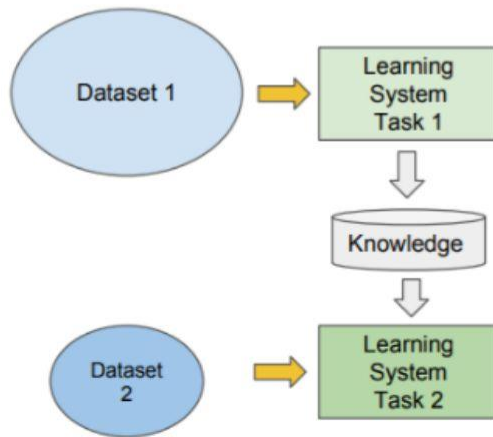
- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



vs

## 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

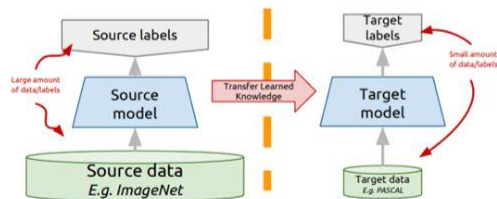
## Transfer learning: idea

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**

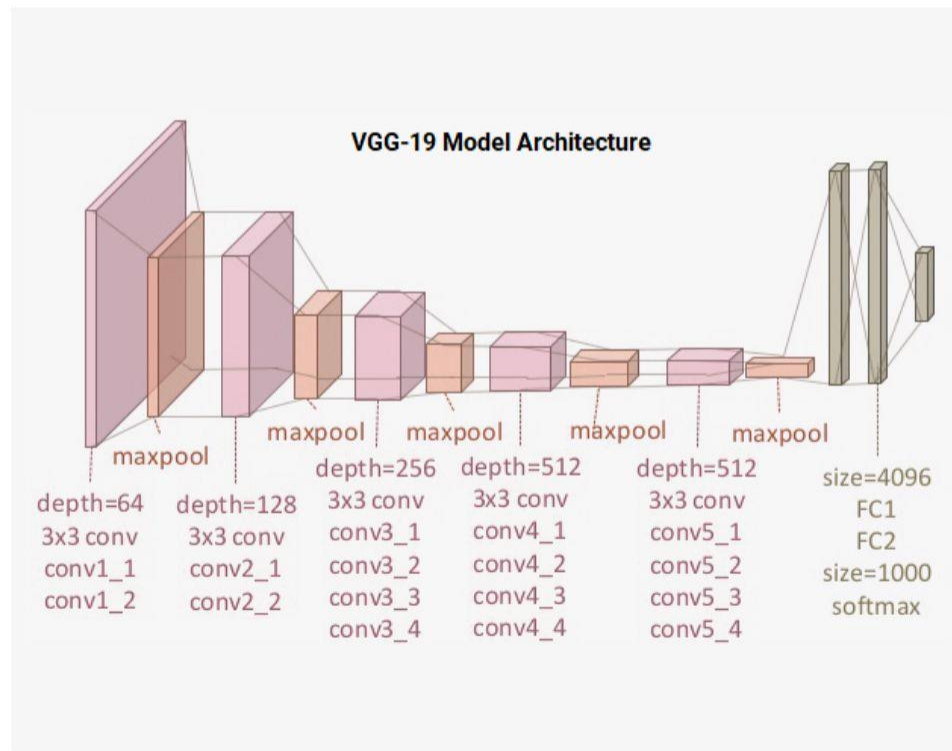
Variations:

- 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



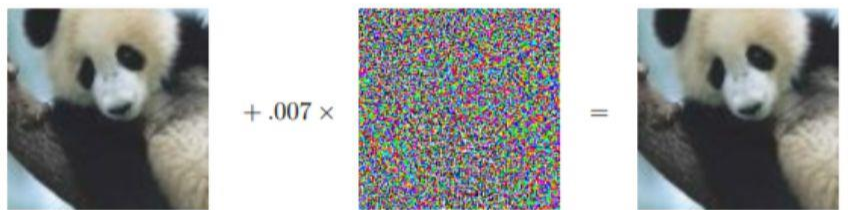
- VGG-19 model is a 19-layer (convolution and fully connected) deep learning network built on the ImageNet database
- 16 convolution layers using 3 x 3 convolution filters along with max pooling layers
- 2 fully connected hidden layers of 4096 units in each layer followed by a dense layer of 1000 units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



# Adversarial Attacks



# Adversarial Attacks



$x$   
"panda"  
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_x J(\theta, x, y))$   
"nematode"  
8.2% confidence

$=$

$x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$   
"gibbon"  
99.3 % confidence

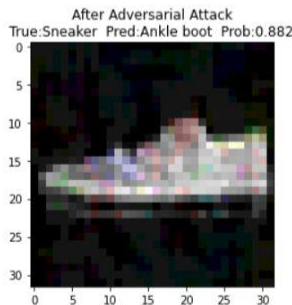
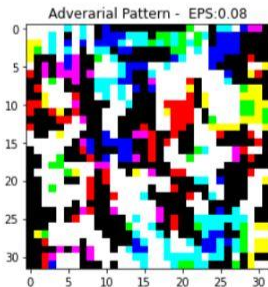
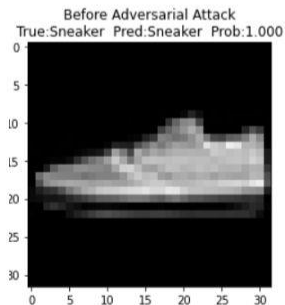
- Purposely adding perturbations or noise in the data to fool the model
- Synthetically created on top of an input image
- Attacker adds small perturbations (distortions) to the original image
- These notorious perturbations are indistinguishable to the human eye, but causes the network to fail

# Adversarial Attacks - Fast Gradient Sign Method

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

where

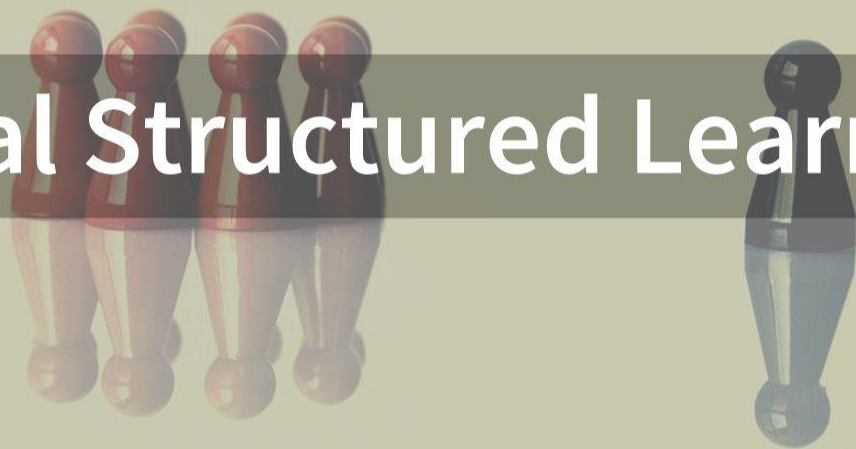
- $adv\_x$  : Adversarial image.
- $x$  : Original input image.
- $y$  : Original input label.
- $\epsilon$  : Multiplier to ensure the perturbations are small.
- $\theta$  : Model parameters.
- $J$  : Loss.



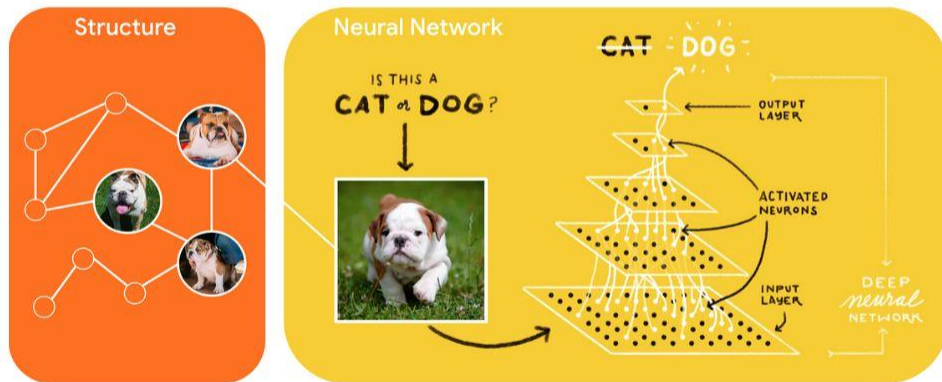
- Uses the gradients of the neural network to create an adversarial example
- Objective is to create an image that maximizes the loss.
- Gradients of the loss with respect to the input image are taken
- A small multiplier (epsilon) is added to the sign of the gradients and added to the original image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



# Neural Structured Learning

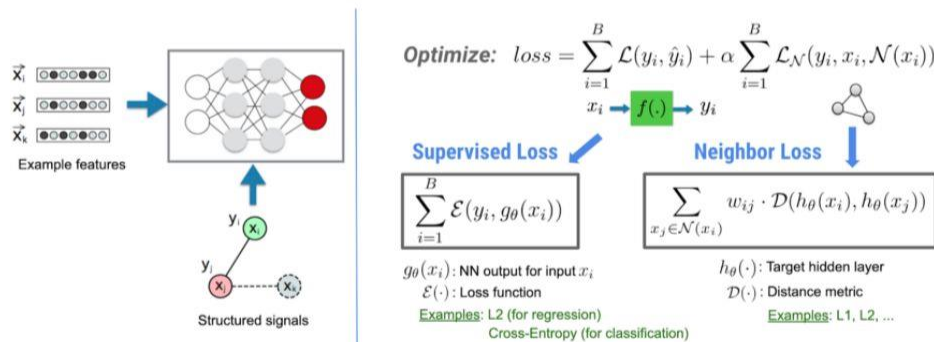


# Neural Structured Learning



- 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

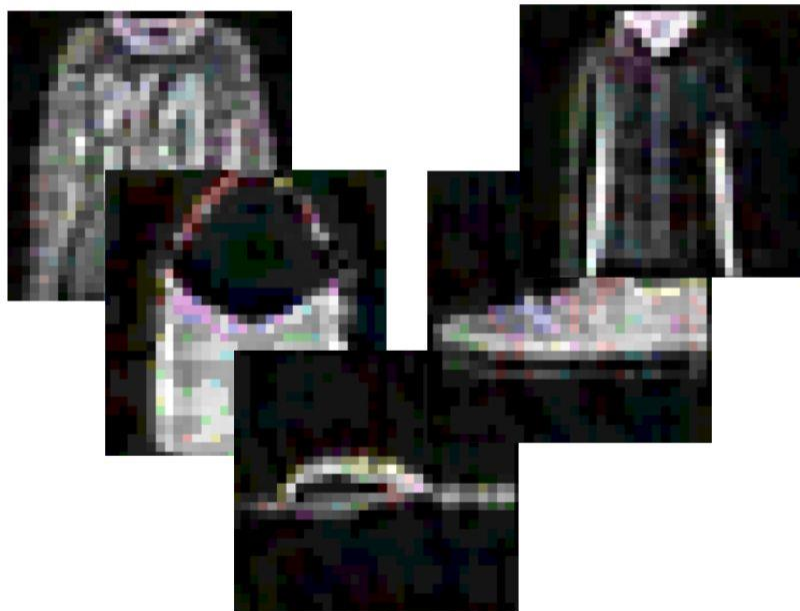
# Neural Structured Learning - Methodology



- 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  
$$\text{total\_loss} = \text{supervised\_loss} + \text{adversarial\_loss}$$
- Minimize supervised loss for accurate predictions
- Minimize adversarial loss to maintain the similarity among inputs and their adversarial examples

# Adversarial Learning with TensorFlow - NSL

```
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
```

} Read  
Data

```
# Create a base model -- sequential, functional, or subclass.
```

```
model = tf.keras.Sequential(...)
```

} Keras  
Model

```
# 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)
```

} Config  
Adv model

```
# 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})
```

} Compile  
Fit  
Eval

A close-up, warm-toned photograph of a person's hands typing on a white Apple keyboard. The keyboard is positioned in front of an iMac, with the Apple logo visible on the back of the monitor. The hands are positioned over the keyboard, with fingers pressing keys. A semi-transparent dark grey rectangular box is overlaid on the center of the image, containing the text "Adversarial Learning Hands-on Tutorial" in white. The background is slightly blurred, showing the desk and the iMac's base.

# Adversarial Learning Hands-on Tutorial

# References

- **Visuals & Content**

- [https://www.tensorflow.org/neural\\_structured\\_learning](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://github.com/dipanjanS/convolutional_neural_networks_essentials)
- [https://www.tensorflow.org/tutorials/generative/adversarial\\_fgsm](https://www.tensorflow.org/tutorials/generative/adversarial_fgsm)
- [https://www.tensorflow.org/neural\\_structured\\_learning/tutorials/adversarial\\_keras\\_cnn\\_mnist](https://www.tensorflow.org/neural_structured_learning/tutorials/adversarial_keras_cnn_mnist)

- **Research Papers**

- [https://github.com/dipanjanS/adversarial\\_learning\\_tfug2020/tree/master/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>