1. Introduction

In this lesson, we're going to go over a few really cool applications of deep learning using pre-trained models that others have generously provided on GitHub. Don't worry if you don't understand what's going on! The goal is just for you to see the power of deep learning in a few different contexts and play around with these models. You'll get to dive deep into such models later on in the program. For now, just have fun and plug in your own examples where possible!

1. Style Transfer

**Style Transfer**

As an example of the kind of things you'll be building with deep learning models, here is a really fun project, [fast style transfer](https://github.com/lengstrom/fast-style-transfer). Style transfer allows you to take famous paintings, and recreate your own images in their styles! The network learns the underlying techniques of those paintings and figures out how to apply them on its own. This model was trained on the styles of famous paintings and is able to transfer those styles to other images and [even videos](https://www.youtube.com/watch?v=xVJwwWQlQ1o)!

I used it to style my cat Chihiro in the style of [Hokusai](https://en.wikipedia.org/wiki/Hokusai)'s [The Great Wave Off Kanagawa](https://en.wikipedia.org/wiki/The_Great_Wave_off_Kanagawa).

[[](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/b0c0431d-f4c6-4628-8f39-8e87e350c294)](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/b0c0431d-f4c6-4628-8f39-8e87e350c294)

To try it out yourself, you can find the code in the [fast-style-transfer GitHub repo](https://github.com/lengstrom/fast-style-transfer). Either use git to clone the repository, or you can download the whole thing as a Zip archive and extract it.

The network has been trained on a few different styles ([here](https://github.com/lengstrom/fast-style-transfer/tree/master/examples/style)) and saved into [checkpoint files](https://drive.google.com/drive/folders/0B9jhaT37ydSyRk9UX0wwX3BpMzQ). Checkpoint files contain all the information about the trained network to apply styles to new images.

**Dependencies**

The easiest way to install all the packages needed to run this code is with [Miniconda](http://conda.pydata.org/miniconda.html" \t "_blank), a smaller version of [Anaconda](https://www.continuum.io/downloads). Miniconda comes with Conda, a package and environment manager built specifically for data science. Install the Python 3 version of Miniconda appropriate for your operating system.

If you haven't used Conda before, please quickly run through the Anaconda lesson (Lesson 2 on this part).

**Windows**

For Windows, you'll need to install TensorFlow, Python 3.5, Pillow 3.4.2, scipy 0.18.1, and numpy 1.11.2. After installing Miniconda, open your command prompt. In there, enter these commands line by line:

conda create -n style-transfer python=3.5

activate style-transfer

pip install tensorflow

conda install scipy pillow

The first line creates a new environment that will hold the packages needed for the style transfer code. The next line (activate style-transfer) enters the environment, you should see the environment name in the prompt at the beginning of the line. The next two install TensorFlow, Scipy, and Pillow (an image processing library).

**OS X and Linux**

For OS X and Linux, you'll need to install TensorFlow 0.11.0, Python 2.7.9, Pillow 3.4.2, scipy 0.18.1, and numpy 1.11.2.

In your terminal, enter this commands line by line:

conda create -n style-transfer python=2.7.9

source activate style-transfer

pip install tensorflow

conda install scipy pillow

The first line creates a new environment that will hold the packages needed for the style transfer code. The next line (source activate style-transfer) enters the environment, , you should see the environment name in the prompt at the beginning of the line. The next two install TensorFlow, Scipy, and Pillow (an image processing library).

**Transferring styles**

1. Download the Zip archive from the [fast-style-transfer](https://github.com/lengstrom/fast-style-transfer) repository and extract it. You can download it by clicking on the bright green button on the right.
2. Download the Rain Princess checkpoint from [here](https://d17h27t6h515a5.cloudfront.net/topher/2017/January/587d1865_rain-princess/rain-princess.ckpt). Put it in the fast-style-transfer folder. A checkpoint file is a model that already has tuned parameters. By using this checkpoint file, we won't need to train the model and can get straight to applying it.
3. Copy the image you want to style into the fast-style-transfer folder (it might be called fast-style-transfer-master).
4. Enter the Conda environment you created above, if you aren't still in it.
5. In your terminal, navigate to the fast-style-transfer folder and enter
6. python evaluate.py --checkpoint ./rain-princess.ckpt --in-path <path\_to\_input\_file> --out-path ./output\_image.jpg

**Note:** Your checkpoint file might be named rain\_princess.ckpt, notice the underscore, it's not the dash from above.

You can get more checkpoint files at the bottom of this page. Try them all!

Share what you create on [Twitter](https://www.twitter.com/) by using the [hashtag #madewithudacity](https://twitter.com/search?f=images&vertical=default&q=%23madewithudacity&src=typd). If you share what you create, your image could be reshared on Udacity's official account! Here's suggested text you can use for your tweet: "Putting deep learning style transfer into practice with @Udacity. #madewithudacity"

Also, feel free to train the network on your own images, you can find instructions in the repository (although it does take some powerful hardware).

**Note:** Be careful with the size of the input image. The style transfer can take quite a while to run on larger images.

Style Transfer Checklist

Task List



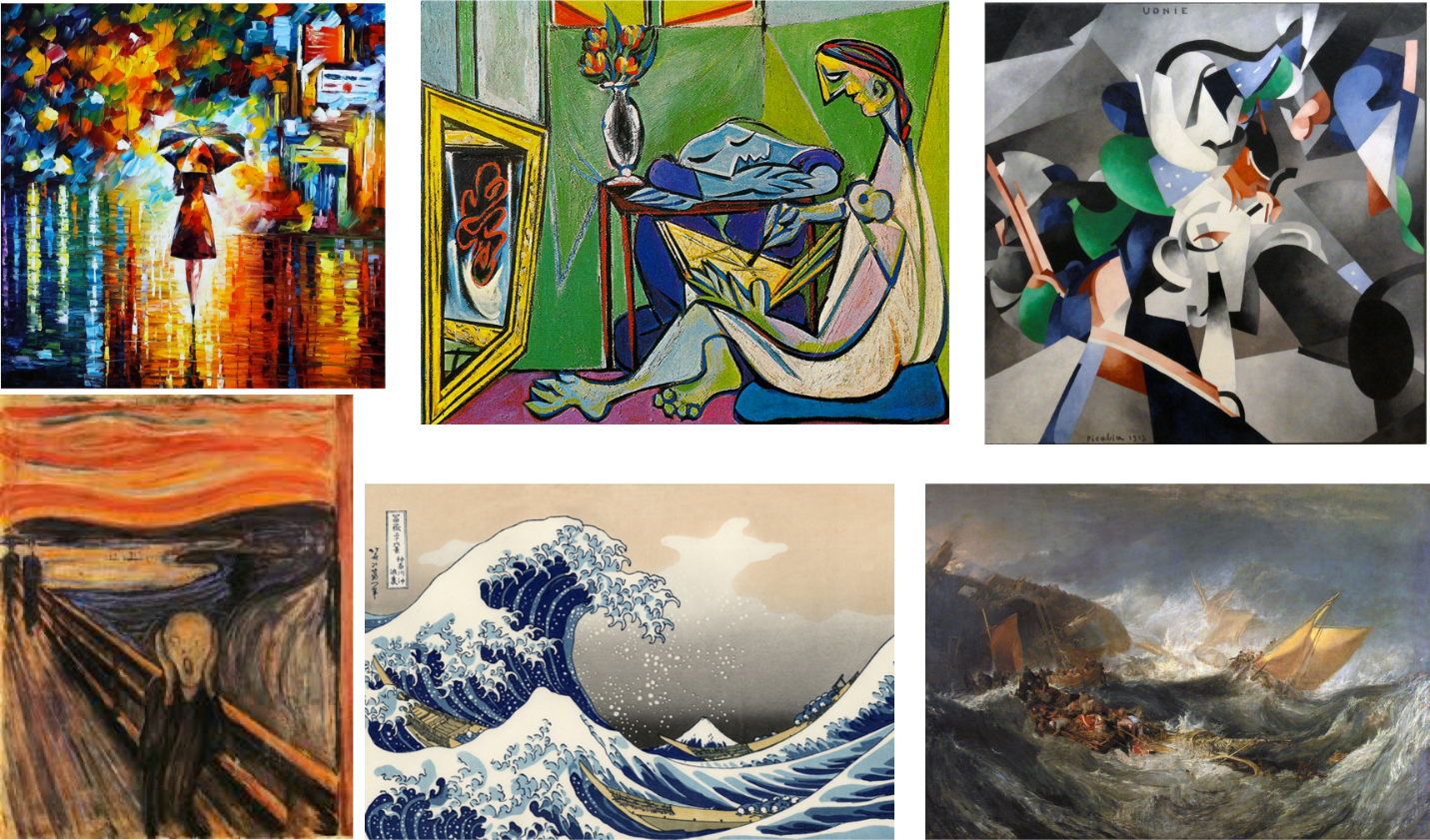
Apply style transfer to an image of yourself or something personal to you.



Share what you create on [Twitter](https://www.twitter.com/) under the [hashtag #madewithudacity](https://twitter.com/search?f=images&vertical=default&q=%23madewithudacity&src=typd).

The checkpoints were trained on the following paintings:

* Rain Princesss, by [Leonid Afremov](https://afremov.com/Leonid-Afremov-bio.html)
* La Muse, by [Pablo Picasso](https://en.wikipedia.org/wiki/Pablo_Picasso)
* Udnie by [Francis Picabia](https://en.wikipedia.org/wiki/Francis_Picabia)
* Scream, by [Edvard Munch](https://en.wikipedia.org/wiki/Edvard_Munch)
* The Great Wave off Kanagawa, by [Hokusai](https://en.wikipedia.org/wiki/Hokusai)
* The Shipwreck of the Minotaur, by [J.M.W. Turner](https://en.wikipedia.org/wiki/J._M._W._Turner)

[[](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/b0c0431d-f4c6-4628-8f39-8e87e350c294)](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/b0c0431d-f4c6-4628-8f39-8e87e350c294)

**Want to know how this works?**

Be sure to enroll in the Deep Learning Nanodegree Program! You’ll have the opportunity to implement the model yourself in your course on Convolutional Neural Networks.

1. DeepTraffic

<https://www.youtube.com/watch?v=az5ElmV4DhY>

Traffic Navigation with Deep Reinforcement Learning

DeepTraffic

Another great application of deep learning is in simulating traffic and making driving decisions. You can find the DeepTraffic simulator here. The network here is attempting to learn a driving strategy such that the car is moving as fast as possible using reinforcement learning. The network is rewarded when the car chooses actions that result in it moving fast. It's this feedback that allows the network to find a strategy of actions for optimal speed.

To learn more about setting the parameters and training the network, read the overview here.

Discuss how you built your network and your results with your fellow students in the #deeptraffic channel on Slack.

1. Flappy Bird

In this example, you'll get to see a deep learning agent playing Flappy Bird! You have the option to train the agent yourself, but for now let's just start with the pre-trained network given by the author. Note that the following agent is able to play without being told any information about the structure of the game or its rules. It automatically discovers the rules of the game by finding out how it did on each iteration.

We will be following [this repository](https://github.com/yenchenlin/DeepLearningFlappyBird) by Yenchen Lin.

[[](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/94e39555-47c6-46ce-92a6-caabedab5ed2)](https://classroom.udacity.com/courses/nd101-preview/lessons/7cb11162-03a5-4ac7-a205-3b72a6bb33b4/concepts/94e39555-47c6-46ce-92a6-caabedab5ed2)

**Instructions**

1. Install miniconda or anaconda if you have not already. You can follow [our tutorial](https://classroom.udacity.com/courses/ud1111?contentVersion=1.0.0&contentLocale=en-us) for help.
2. Create an environment for flappybird
   * Mac/Linux: conda create --name=flappybird python=2.7
   * Windows: conda create --name=flappybird python=3.5
3. Enter your conda environment
   * Mac/Linux: source activate flappybird
   * Windows: activate flappybird
4. conda install -c menpo opencv3
5. pip install pygame
6. pip install tensorflow
7. git clone https://github.com/yenchenlin/DeepLearningFlappyBird.git or download it as a ZIP archive from [the repo](https://github.com/yenchenlin/DeepLearningFlappyBird).
8. cd DeepLearningFlappyBird (If you downloaded the ZIP archive, it might be called DeepLearningFlappyBird-master)
9. python deep\_q\_network.py

If all went correctly, you should be seeing a deep learning based agent play Flappy Bird! The repository contains instructions for training your own agent if you're interested!

