

Extracurricular Computer Vision Nangodegr

Applications of computer vision	2
Style Transfer.....	2
DeepTraffic.....	5
Flappy Bird	5
Instructions.....	6
Books to read.....	7
Review: Training A Neural Network	7
Lesson 1: Feedforward and back propagation.....	7
Neural Network Architecture.....	8
Feedforward	9
Error Function.....	9
Backpropagation.....	9
Chain Rule	10
Lesson 2: Training Neural Networks.....	10
Overfitting and Underfitting	11
Dropout.....	11
Lesson 3: Deep Learning with pytorch	12
Skin Cancer Detection.....	14
Lesson 1	14
Validating the Training.....	16
Confusion Matrices	20
Conclusion:.....	21
Useful Resources	22
Mini Project:	23
More Deep Learning Models	29
Lesson 1	29
Methods of upsampling.....	30
Semantic Segmentation.....	31
IoU	33
FCN-8, Architecture.....	35
Text Sentiment Analysis.....	37
Lesson 1	37
Mini Project 1	38
Instructions-----	39

Extracurricular Computer Vision Nangodegr

Building a Neural Network.....	40
Understanding Neural Noise.....	41
Understanding Inefficiencies in our Network	41
Conclusion:.....	43

Applications of computer vision

LESSON 1

Applying Deep Learning Models

Try out a few really cool applications of computer vision and deep learning, such as style transfer, using pre-trained models that others have generously provided on Github.



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](#). 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](#)!

Mat Leonard, the product lead for the school of AI, used it to style his cat Chihiro in the style of [Hokusai's The Great Wave Off Kanagawa](#).

Extracurricular Computer Vision Nangodegr



Mat's cat in the style of Hokusai.

To try it out yourself, you can find the code in the [fast-style-transfer GitHub repo](#). 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](#)) and saved into [checkpoint files](#).

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](#), a smaller version of [Anaconda](#). 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 3 on this part).

Windows

For Windows, you'll need to install TensorFlow 0.12.1, 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
activate style-transfer
conda install tensorflow scipy pillow
pip install moviepy
```

Extracurricular Computer Vision Nangodegr

```
python -c "import imageio; imageio.plugins.ffmpeg.download()"
```

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=3
activate style-transfer
conda install tensorflow scipy pillow
pip install moviepy
python -c "import imageio; imageio.plugins.ffmpeg.download()"
```

Let's take a quick look at what these commands do. The first line in both sets of instructions, creates a new environment with Python 3. This environment will hold all the packages you need for the style transfer code. The next line enters the environment. Next, we install TensorFlow, SciPy, Pillow (which is an image processing library), and moviepy. The last line here installs ffmpeg, an application for converting images and videos.

Transferring styles

- Download the Zip archive from (or clone) the [fast-style-transfer](#) repository and extract it. You can download it by clicking on the bright green button on the right.
- Download the Rain Princess checkpoint from [here](#). 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.
- Copy the image you want to style into the fast-style-transfer folder.
- Enter the Conda environment you created above, if you aren't still in it.
- Finally, in your terminal, navigate to the fast-style-transfer folder and enter

```
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.

Extracurricular Computer Vision Nangodegr

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

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

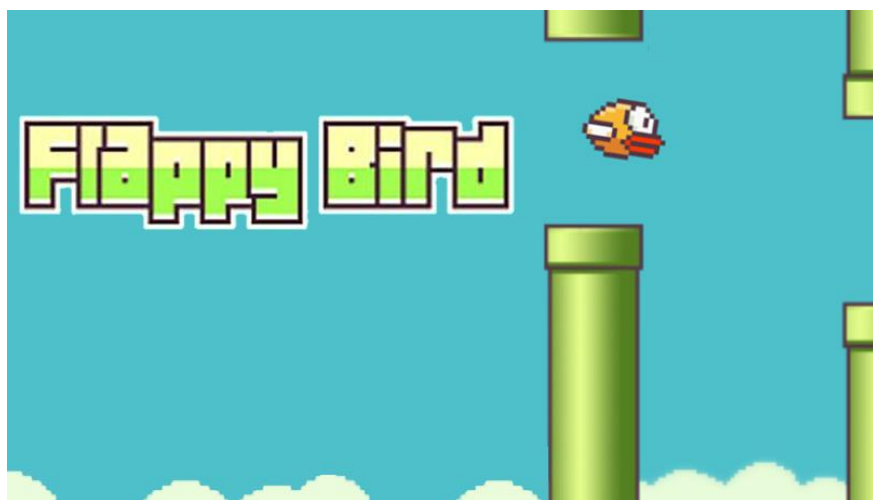
DeepTraffic

https://www.youtube.com/watch?time_continue=10&v=az5ElmV4DhY&feature=emb_logo

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](#) by Yenchen Lin.



Extracurricular Computer Vision Nangodegr

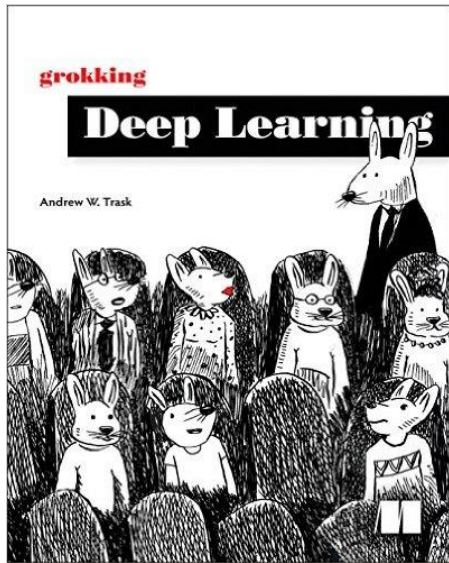
Instructions

- Install miniconda or anaconda if you have not already.
- Create an environment for flappybird
 - Mac/Linux: `conda create --name=flappybird python=2.7`
 - Windows: `conda create --name=flappybird python=3.5`
- Enter your conda environment: `conda activate flappybird`
- `conda install opencv`
 - If you encounter an error here, you may try an **alternate** download path and *instead* type `conda install --channel https://conda.anaconda.org/menpo opencv3`
- `pip install pygame`
- `pip install tensorflow==0.12`
- `git clone https://github.com/yenchenlin/DeepLearningFlappyBird.git`
 - If you don't have git installed, you can download and extract the zip archive directly from [the repository](#)
- `cd DeepLearningFlappyBird`
 - If you downloaded the archive, you will need to navigate to the extracted folder **DeepLearningFlappyBird-master** instead
- `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!

You can also, typically, force-quit out of the game by returning to your terminal window and typing `Command or Ctrl + C`.

Extracurricular Computer Vision Nanodegr



Books to read

We believe that you learn best when you are exposed to multiple perspectives on the same idea. As such, we recommend checking out a few of the books below to get an added perspective on Deep Learning.

- [Grokking Deep Learning](#) by Andrew Trask. Use our exclusive discount code **traskud17** for 40% off. This provides a very gentle introduction to Deep Learning and covers the intuition more than the theory.
- [Neural Networks And Deep Learning](#) by Michael Nielsen. This book is more rigorous than Grokking Deep Learning and includes a lot of fun, interactive visualizations to play with.
- [The Deep Learning Textbook](#) from Ian Goodfellow, Yoshua Bengio, and Aaron Courville. This online book contains a lot of material and is the most rigorous of the three books suggested.

Review: Training A Neural Network

Lesson 1: Feedforward and back propagation

Extracurricular Computer Vision Nanodegree

LESSON 1

Feedforward and Backpropagation

Short introduction to neural networks: how they train by doing a feedforward pass then performing backpropagation.

CONTINUE →



Why "Neural Networks"?

https://www.youtube.com/watch?v=zAkzOZntK6Y&feature=emb_logo

Neural Network Architecture

Ok, so we're ready to put these building blocks together, and build great Neural Networks! (Or Multi-Layer Perceptrons, however you prefer to call them.)

This first two videos will show us how to combine two perceptrons into a third, more complicated one.

https://www.youtube.com/watch?v=Boy3zHVrWB4&feature=emb_logo

https://www.youtube.com/watch?v=FWN3Sw5fFoM&feature=emb_logo

QUESTION 1 OF 2

Based on the above video, let's define the combination of two new perceptrons as $w_1 * 0.4 + w_2 * 0.6 + b$. Which of the following values for the weights and the bias would result in the final probability of the point to be 0.88?

☐ $w_1: 2, w_2: 6, b: -2$

☒ $w_1: 3, w_2: 5, b: -2.2$

Multiple layers

Now, not all neural networks look like the one above. They can be way more complicated! In particular, we can do the following things:

- Add more nodes to the input, hidden, and output layers.

Extracurricular Computer Vision Nangodegr

- Add more layers.

We'll see the effects of these changes in the next video.

https://www.youtube.com/watch?v=pg99FkXYKOM&feature=emb_logo

Multi-Class Classification

And here we elaborate a bit more into what can be done if our neural network needs to model data with more than one output.

https://www.youtube.com/watch?v=uNTtvxwfox0&feature=emb_logo

How many nodes in the output layer would you require if you were trying to classify all the letters in the English alphabet?

Feedforward

Feedforward is the process neural networks use to turn the input into an output. Let's study it more carefully, before we dive into how to train the networks.

https://www.youtube.com/watch?v=hVCuvMGOfyY&feature=emb_logo

Error Function

Just as before, neural networks will produce an error function, which at the end, is what we'll be minimizing. The following video shows the error function for a neural network.

https://www.youtube.com/watch?v=SC1wEW7TtKs&feature=emb_logo

Backpropagation

Now, we're ready to get our hands into training a neural network. For this, we'll use the method known as **backpropagation**. In a nutshell, backpropagation will consist of:

- Doing a feedforward operation.
 - Comparing the output of the model with the desired output.
 - Calculating the error.
 - Running the feedforward operation backwards (backpropagation) to spread the error to each of the weights.
-

Extracurricular Computer Vision Nangodegr

- Use this to update the weights, and get a better model.
- Continue this until we have a model that is good.

Sounds more complicated than what it actually is. Let's take a look in the next few videos. The first video will show us a conceptual interpretation of what backpropagation is.

https://www.youtube.com/watch?v=1SmY3TZTyUk&feature=emb_logo

Backpropagation Math

And the next few videos will go deeper into the math. Feel free to tune out, since this part gets handled by Keras pretty well. If you'd like to go start training networks right away, go to the next section. But if you enjoy calculating lots of derivatives, let's dive in!

In the video below at 1:24, the edges should be directed to the sigmoid function and not the bias at that last layer; the edges of the last layer point to the bias currently which is incorrect.

https://www.youtube.com/watch?v=tVuZDbUrzzl&feature=emb_logo

Chain Rule

We'll need to recall the chain rule to help us calculate derivatives.

https://www.youtube.com/watch?v=YAhlBOnbt54&feature=emb_logo

https://www.youtube.com/watch?v=7lidiTGIlN4&feature=emb_logo

Calculation of the derivative of the sigmoid function

Recall that the sigmoid function has a beautiful derivative, which we can see in the following calculation. This will make our backpropagation step much cleaner.

$$\begin{aligned}
 \sigma'(x) &= \frac{\partial}{\partial x} \frac{1}{1+e^{-x}} \\
 &= \frac{e^{-x}}{(1+e^{-x})^2} \\
 &= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
 &= \sigma(x)(1 - \sigma(x))
 \end{aligned}$$

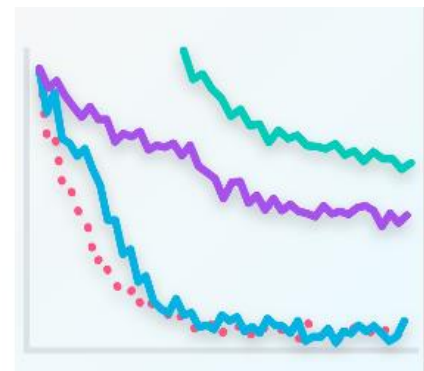
Lesson 2: Training Neural Networks

LESSON 2

Training Neural Networks

Now that you know what neural networks are, in this lesson you will learn several techniques to improve their training.

CONTINUE →



https://www.youtube.com/watch?v=UiGKhx9pUYc&feature=emb_logo

Extracurricular Computer Vision Nangodegr

https://www.youtube.com/watch?v=EeBZpb-PSac&feature=emb_logo

Overfitting and Underfitting

https://www.youtube.com/watch?v=xj4PIXMsN-Y&feature=emb_logo

https://www.youtube.com/watch?v=NnS0FJyVcDQ&feature=emb_logo

Regularization

https://www.youtube.com/watch?v=KxROxcRsHL8&feature=emb_logo

Quiz: Which gives a smaller error?

Prediction: $\hat{y} = \sigma(w_1x_1 + w_2x_2 + b)$

☐ **Solution 1:** $x_1 + x_2$

☐ **Solution 2:** $10x_1 + 10x_2$

QUIZ QUESTION

Which gives a smaller error?

☐ $x_1 + x_2$

☒ $10x_1 + 10x_2$

https://www.youtube.com/watch?v=ndYnUrx8xvs&feature=emb_logo

Dropout

https://www.youtube.com/watch?time_continue=6&v=Ty6K6YiGdBs&feature=emb_logo

Extracurricular Computer Vision Nanodegr

Local Minima

https://www.youtube.com/watch?v=gF_sW_nY-xw&feature=emb_logo

https://www.youtube.com/watch?v=idyBBCzXiagg&feature=emb_logo

Vanishing Gradient

https://www.youtube.com/watch?time_continue=3&v=W_JJm_5syFw&feature=emb_logo

Other Activation Functions

https://www.youtube.com/watch?time_continue=3&v=kA-1vUt6cvQ&feature=emb_logo

Batch vs Stochastic Gradient Descent

https://www.youtube.com/watch?time_continue=76&v=2p58rVgqsgo&feature=emb_logo

Learning Rate Decay

https://www.youtube.com/watch?time_continue=17&v=TwJ8aSZoh2U&feature=emb_logo

https://www.youtube.com/watch?time_continue=6&v=r-rYz_PEWc8&feature=emb_logo

Error Functions Around the World

https://www.youtube.com/watch?time_continue=1&v=34AAcTECu2A&feature=emb_logo

Lesson 3: Deep Learning with pytorch

LESSON 3

Deep Learning with PyTorch

Learn how to use PyTorch for building deep learning models



Extracurricular Computer Vision Nangodegr

Lesson Overview

Welcome! In this lesson, you'll learn how to use PyTorch for building deep learning models. PyTorch was released in early 2017 and has been making a pretty big impact in the deep learning community. It's developed as an open source project by the [Facebook AI Research team](#), but is being adopted by teams everywhere in industry and academia. In my experience, it's the best framework for learning deep learning and just a delight to work with in general. By the end of this lesson, you'll have trained your own deep learning model that can classify images of cats and dogs.

I'll first give you a basic introduction to PyTorch, where we'll cover **tensors** - the main data structure of PyTorch. I'll show you how to create tensors, how to do simple operations, and how tensors interact with NumPy.

Then you'll learn about a module called **autograd** that PyTorch uses to calculate gradients for training neural networks. Autograd, in my opinion, is amazing. It does all the work of backpropagation for you by calculating the gradients at each operation in the network which you can then use to update the network weights.

Next you'll use PyTorch to build a network and run data forward through it. After that, you'll define a loss and an optimization method to train the neural network on a dataset of handwritten digits. You'll also learn how to test that your network is able to generalize through **validation**.

However, you'll find that your network doesn't work too well with more complex images. You'll learn how to use pre-trained networks to improve the performance of your classifier, a technique known as **transfer learning**.

If you'd like to work through the notebooks on your own machine or otherwise outside the classroom, you can find the code in [this repo](#).

See you in the lesson!

Pytorch Tensors:

https://www.youtube.com/watch?v=n4mbZYIfKb4&feature=emb_logo

Defining Network

https://www.youtube.com/watch?v=u50_ZyKqt8g&feature=emb_logo

Training Networks:

https://www.youtube.com/watch?v=u8hDj5aJK6l&feature=emb_logo

Fashion-MNIST Exercise

https://www.youtube.com/watch?v=AEJV_RKZ7VU&feature=emb_logo

Inference & Validation

Extracurricular Computer Vision Nangodegr

https://www.youtube.com/watch?v=coBbbrGZXI0&feature=emb_logo

Saving and Loading Trained Networks

https://www.youtube.com/watch?v=HiTih59dCWQ&feature=emb_logo

Loading Data Sets with Torchvision

https://www.youtube.com/watch?v=hFu7GTfRWks&feature=emb_logo

Transfer Learning

https://www.youtube.com/watch?v=3eqn5sgCOsY&feature=emb_logo

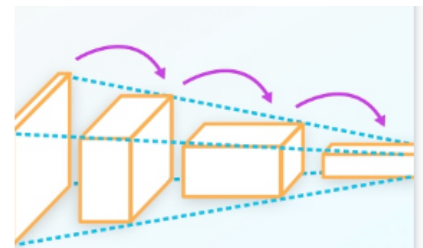
Skin Cancer Detection

Lesson 1

Deep Learning for Cancer Detection with Sebastian Thrun

Sebastian Thrun teaches us about his groundbreaking work detecting skin cancer with convolutional neural networks.

[https://www.youtube.com/watch?v=ZCpXvVldnY&feature=emb_logo](#)



https://www.youtube.com/watch?v=ZCpXvVldnY&feature=emb_logo

Skin Cancer:

https://www.youtube.com/watch?v=70jGZeITNgk&feature=emb_logo

Survival Probability of Skin Cancer

https://www.youtube.com/watch?time_continue=6&v=QPlp3NeGuSk&feature=emb_logo

Medical Classification

Extracurricular Computer Vision Nangodegr

https://www.youtube.com/watch?time_continue=2&v=RCOSP60dV7U&feature=emb_logo

The data

https://www.youtube.com/watch?v=2RLbbV7MQNA&feature=emb_logo

Challenge

https://www.youtube.com/watch?v=Efnoj1KNPHw&feature=emb_logo

Looking at the following images, could you tell the characteristics that determine if a lesion is benign (above) or malignant (below)?



https://www.youtube.com/watch?v=F8yc7BIV93c&feature=emb_logo

https://www.youtube.com/watch?v=1z3o4niQuNg&feature=emb_logo

Extracurricular Computer Vision Nanodegr

Training: https://www.youtube.com/watch?v=Hwil-UXUx-M&feature=emb_logo

https://www.youtube.com/watch?v=DRC1e4XGI2M&feature=emb_logo





https://www.youtube.com/watch?v=sOuoRZRKDzs&feature=emb_logo

Validating the Training

https://www.youtube.com/watch?time_continue=5&v=Oxm9ofvov3l&feature=emb_logo

https://www.youtube.com/watch?v=O17MnhWBmKA&feature=emb_logo

https://www.youtube.com/watch?v=GBZjyeMjKxc&feature=emb_logo

	Diagnosed Sick	Diagnosed Healthy	
Sick	 True Positive	 False Negative	Recall
Healthy	 False Positive	 True Negative	
	Precision		

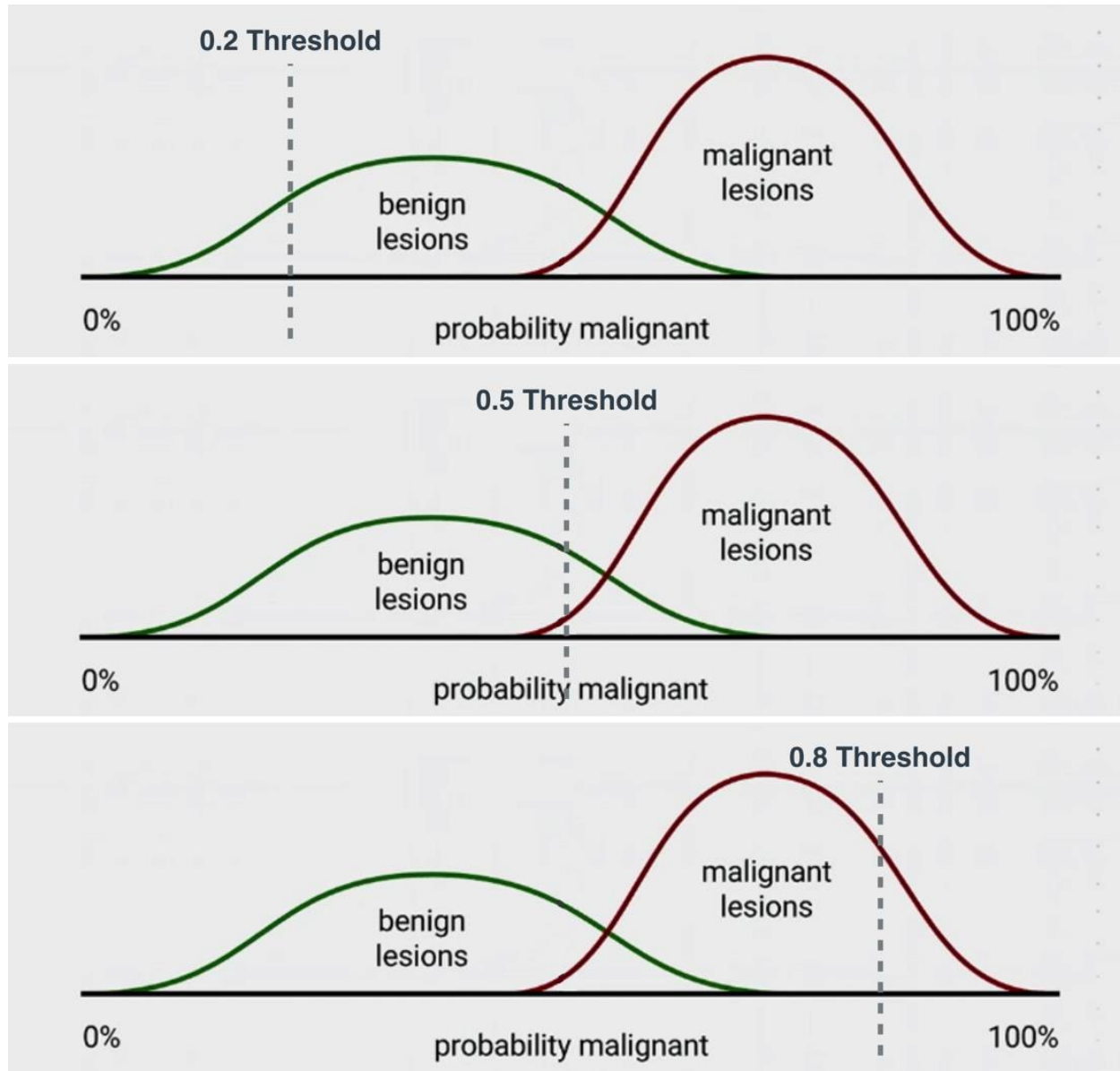
https://www.youtube.com/watch?v=4UzkwecBJro&feature=emb_logo

https://www.youtube.com/watch?v=IJYvt2ssUFk&feature=emb_logo

The graph below is a histogram of the predictions our model gives in a set of images of lesions, as follows:

Extracurricular Computer Vision Nangodegr

- Each point in the horizontal axis is a value ppp from 0 to 1.
- Over each value ppp , we locate all the lesions that our classifier predicted to have probability p of being malignant.



Here we have graphed the thresholds at 0.2, 0.5, and 0.8. Notice how:

- At 0.2, we classify every malignant lesion correctly, yet we also send a lot of benign lesions for more testing.
- At 0.5, we miss some malignant lesions (bad), and we send a few benign lesions for more testing.

Extracurricular Computer Vision Nangodegr

- At 0.8, we correctly classify most of the benign lesions, but we miss many malignant lesions (very bad).

So in this case, it's arguable that 0.2 is better.

However, for this model, there may even be a better value for the threshold. What would it be?

Quiz Question

What would be the perfect value for the threshold in this model?

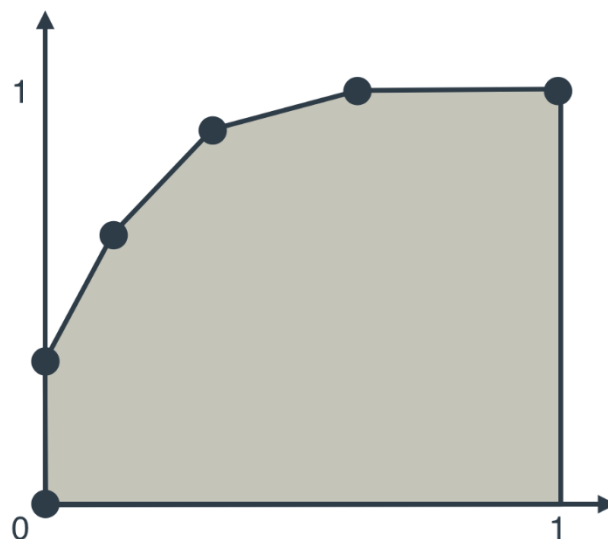
ROC Curves

If you'd like to refresh your memory on ROC curves, check out the below video from Luis.

https://www.youtube.com/watch?v=2lw5TiGzJl4&feature=emb_logo

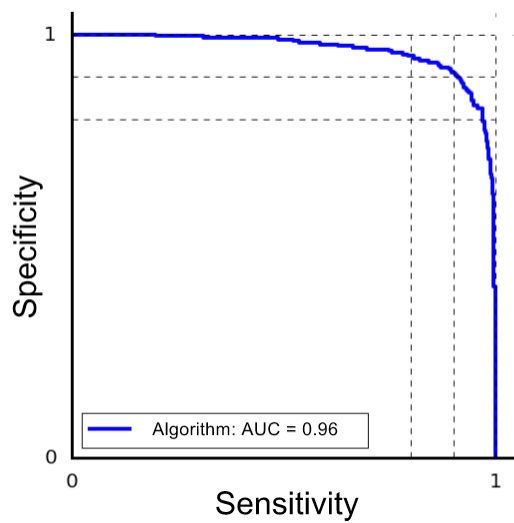
The curves have been introduced as follows, where in the horizontal axis we plot the True Positive Rate, and in the vertical axis we plot the False Positive Rate.

ROC Curve



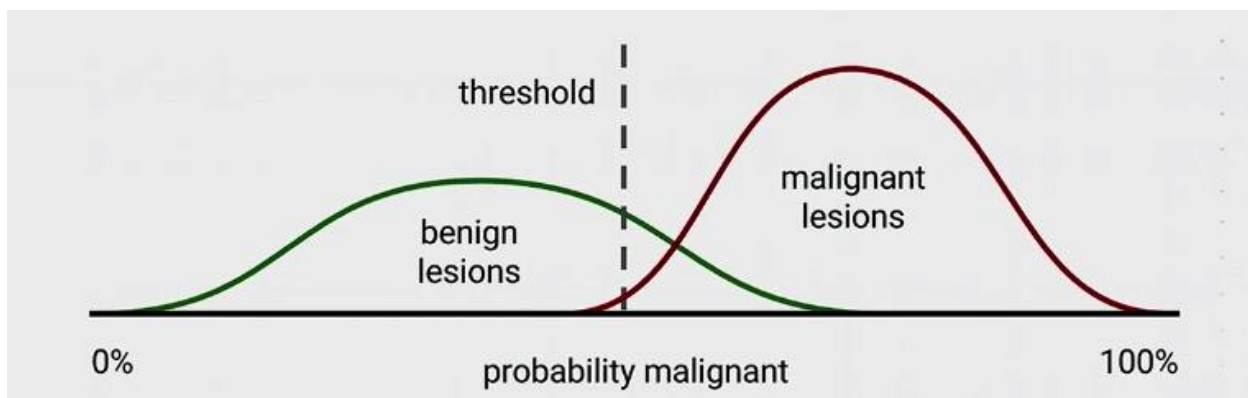
However, you'll see that in this section, I will use a different ROC Curve. The one I use looks like I flipped it sideways, like this:

Extracurricular Computer Vision Nangodegr



And there's a really cool reason why I use this one. And it's because it's the curve we get when we plot the sensitivity in the horizontal axis, and the specificity in the vertical axis!

Let me be more specific (yes pun intended). Let's use the same histogram as in the last section.



Recall that the values in the horizontal axis are all the possible thresholds. For any threshold ppp between 0 and 1, the verdict of the model will be the following: "Any lesion to the left of this threshold will be considered benign, and any lesion to the right of this threshold will be considered malignant, and sent for more tests."

Now, for this particular model, we calculate the sensitivity and specificity as follows:

- Sensitivity: Out of all the malignant lesions, what percentage are to the right of the threshold (correctly classified)?

Extracurricular Computer Vision Nangodegr

- **Specificity:** Out of all the benign lesions, what percentage are to the left of the threshold (correctly classified)?

And we plot that point, where the coordinates are (Sensitivity, Specificity). If we plot all the points corresponding to each of the possible thresholds between 0% and 100%, we'll get the ROC curve that I drew above. Therefore, we can also refer to the ROC curve as the *Sensitivity-Specificity Curve*.

And finally, here's a little animation of the ROC curve getting drawn, as the threshold moves from 0 to

https://www.youtube.com/watch?v=1GdiN5Wc8LA&feature=emb_logo

https://www.youtube.com/watch?v=Xv3v59_CfEU&feature=emb_logo

https://www.youtube.com/watch?v=sdUUf6RRmXI&feature=emb_logo

Comparing our Results with Doctors

https://www.youtube.com/watch?time_continue=3&v=fWwe_JlpnlQ&feature=emb_logo

Visualization

https://www.youtube.com/watch?time_continue=4&v=aGIGB4Ta3_A&feature=emb_logo

What is the network looking at?

https://www.youtube.com/watch?time_continue=18&v=qN-rvoxPbBw&feature=emb_logo

Confusion Matrices

In Luis's Evaluation Metrics section, we learned about confusion matrices, and if you need a refresher, the video is below.

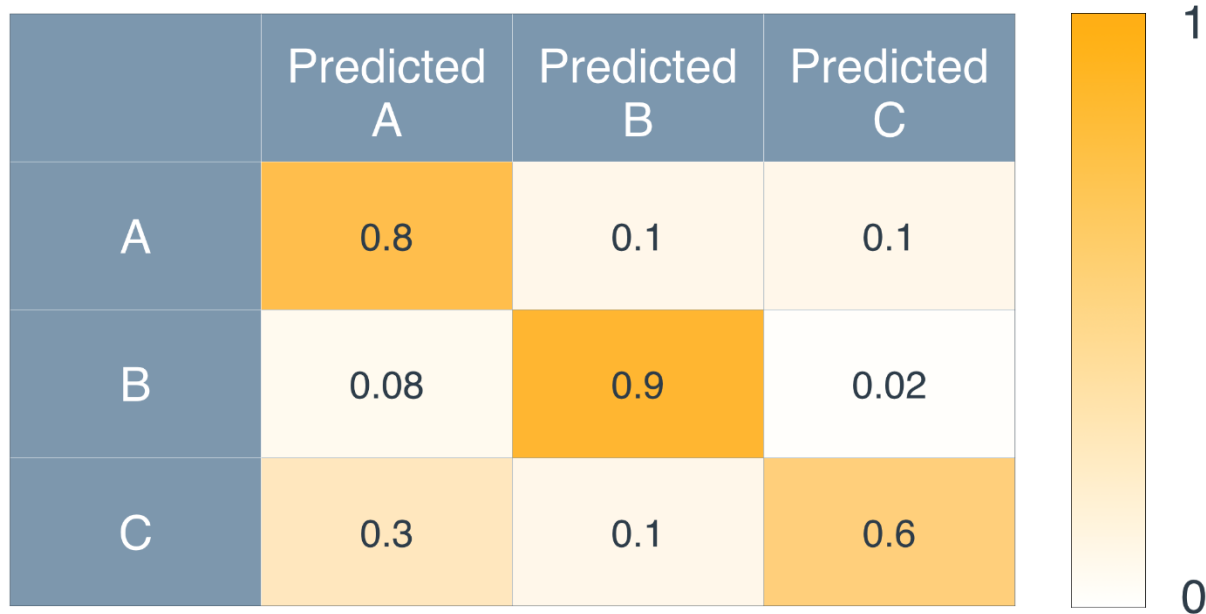
Type 1 and Type 2 Errors

Sometimes in the literature, you'll see False Positives and True Negatives as Type 1 and Type 2 errors. Here is the correspondence:

- **Type 1 Error (Error of the first kind, or False Positive):** In the medical example, this is when we misdiagnose a healthy patient as sick.
- **Type 2 Error (Error of the second kind, or False Negative):** In the medical example, this is when we misdiagnose a sick patient as healthy.

But confusion matrices can be much larger than 2×2 . Here's an example of a larger one. Let's say we have three illnesses called A, B, C. And here is a confusion matrix:

Extracurricular Computer Vision Nangodegr



A confusion matrix for three types of illnesses: A, B, and C

As you can see, each entry in the i i -th row and the j j -th column will tell you the probability of the patient having illness i i and getting diagnosed with illness j j .

For example, from the entry on the second row and the first column, we can determine that if a patient has illness B, the probability of getting diagnosed with illness A is exactly 0.08.

https://www.youtube.com/watch?v=9GLNjmMUB_4&feature=emb_logo

https://www.youtube.com/watch?v=3rpN-YYIfes&feature=emb_logo

Conclusion:

https://www.youtube.com/watch?time_continue=4&v=WhpE_8sTt-0&feature=emb_logo

Extracurricular Computer Vision Nangodegr

Useful Resources



Here's our publication in [Nature](#).

Other articles about our work:

- [Fortune Magazine](#)
- [Bloomberg](#)
- [BBC](#)
- [Wall Street Journal](#)
- [Forbes](#)
- [Scientific American](#)

Extracurricular Computer Vision Nanodegree

Mini Project:

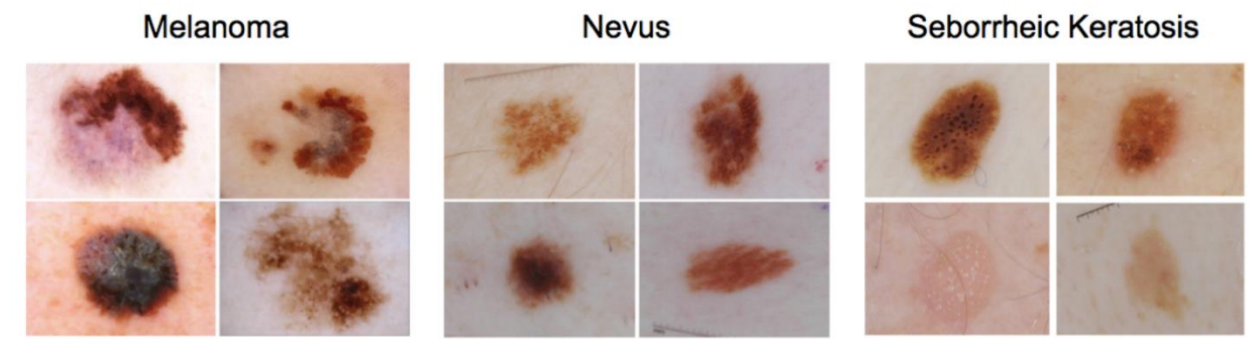
https://www.youtube.com/watch?v=Rgf3YVFWI-M&feature=emb_logo

Mini Project: Dermatologist AI

Introduction

In this mini project, you will design an algorithm that can visually diagnose **melanoma**, the deadliest form of skin cancer. In particular, your algorithm will distinguish this malignant skin tumor from two types of benign lesions (**nevi** and **seborrheic keratoses**).

The data and objective are pulled from the **2017 ISIC Challenge on Skin Lesion Analysis Towards Melanoma Detection**. As part of the challenge, participants were tasked to design an algorithm to diagnose skin lesion images as one of three different skin diseases (melanoma, nevus, or seborrheic keratosis). In this project, you will create a model to generate your own predictions.



Getting Started

1. Clone the **repository** and create a data/ folder to hold the dataset of skin images.
 2. git clone <https://github.com/udacity/dermatologist-ai.git>
 3. mkdir data; cd data
 4. Create folders to hold the training, validation, and test images.
 5. mkdir train; mkdir valid; mkdir test
 6. Download and unzip the **training data** (5.3 GB).
 7. Download and unzip the **validation data** (824.5 MB).
 8. Download and unzip the **test data** (5.1 GB).
-

Extracurricular Computer Vision Nanodegr

9. Place the training, validation, and test images in the data/ folder, at data/train/, data/valid/, and data/test/, respectively. Each folder should contain three sub-folders (melanoma/, nevus/, seborrheic_keratosis/), each containing representative images from one of the three image classes.

You are free to use any coding environment of your choice to solve this mini project! In order to rank your results, you need only use a pipeline that culminates in a CSV file containing your test predictions.

Create a Model

Use the training and validation data to train a model that can distinguish between the three different image classes. (After training, you will use the test images to gauge the performance of your model.)

If you would like to read more about some of the algorithms that were successful in this competition, please read [this article](#) that discusses some of the best approaches. A few of the corresponding research papers appear below.

- Matsunaga K, Hamada A, Minagawa A, Koga H. [“Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble”](#). International Skin Imaging Collaboration (ISIC) 2017 Challenge at the International Symposium on Biomedical Imaging (ISBI).
- Daz IG. [“Incorporating the Knowledge of Dermatologists to Convolutional Neural Networks for the Diagnosis of Skin Lesions”](#). International Skin Imaging Collaboration (ISIC) 2017 Challenge at the International Symposium on Biomedical Imaging (ISBI). ([github](#))
- Menegola A, Tavares J, Fornaciali M, Li LT, Avila S, Valle E. [“RECOD Titans at ISIC Challenge 2017”](#). International Skin Imaging Collaboration (ISIC) 2017 Challenge at the International Symposium on Biomedical Imaging (ISBI). ([github](#))

While the original challenge provided additional data (such as the gender and age of the patients), we only provide the image data to you. If you would like to download this additional patient data, you may do so at the competition [website](#).

All three of the above teams increased the number of images in the training set with additional data sources. If you'd like to expand your training set, you are encouraged to begin with the [ISIC Archive](#).

Evaluation

Inspired by the ISIC challenge, your algorithm will be ranked according to three separate categories.

Category 1: ROC AUC for Melanoma Classification

In the first category, we will gauge the ability of your CNN to distinguish between malignant melanoma and the benign skin lesions (nevus, seborrheic keratosis) by calculating the area under the receiver operating characteristic curve ([ROC AUC](#)) corresponding to this binary classification task.

If you are unfamiliar with ROC (Receiver Operating Characteristic) curves and would like to learn more, you can check out the documentation in [scikit-learn](#) or read [this Wikipedia article](#).

The top scores (from the ISIC competition) in this category can be found in the image below.

Extracurricular Computer Vision Nangodegr

Rank	User	Title	Organization	Documentation	Date	Score
1	RECOD Titans	release (rc36xtrm) "alea jacta est"	RECOD Titans / UNICAMP	[link]	Wed, 1 Mar 2017, 11:42:07 pm	0.874
2	Lei Bi	EResNet (single scale w/o attributes)	USYD-BMIT	[link]	Wed, 1 Mar 2017, 8:04:42 pm	0.870
3	Kazuhisa Matsunaga	ResNet ensemble with normalized image	Casio and Shinshu University joint team	[link]	Wed, 1 Mar 2017, 11:18:03 pm	0.868
4	monty python	gpm-LSSSD	Multimedia Processing Group - Universidad Carlos III de Madrid	[link]	Wed, 1 Mar 2017, 12:57:35 pm	0.856
5	T D	Last Minute Submission!!!!	University of Guelph - MLRG	[link]	Wed, 1 Mar 2017, 11:55:50 pm	0.836
6	Xulei Yang	multi-task deep learning model for skin lesion segmentation and classification-3	Institute of High Performance Computing + National Skin Center, Singapore	[link]	Tue, 28 Feb 2017, 6:34:10 pm	0.830
7	Rafael Sousa	Araguaia Medical Vision Lab - GooglAlexNet	Universidade Federal de Mato Grosso	[link]	Wed, 1 Mar 2017, 3:26:22 pm	0.805
8	x j	finalv_L2C1_trir	CVI	[link]	Wed, 1 Mar 2017, 11:17:56 am	0.804
9	Cristina Vasconcelos	comb	icuff	[link]	Tue, 28 Feb 2017, 1:11:21 am	0.791
10	C V	all	icuff	[link]	Tue, 28 Feb 2017, 1:06:44 am	0.789
11	Euljoon Ahn	DeepAhn	USYD-BMIT	[link]	Wed, 1 Mar 2017, 10:30:13 am	0.786
12	Balázs Harangi	Ensemble of deep convolutional neural networks	University of Debrecen	[link]	Wed, 1 Mar 2017, 8:25:16 pm	0.783
13	Matt Berseth	Final Classification Submission	NLPLOGIX / WISEEYE.AI	[link]	Tue, 28 Feb 2017, 6:32:47 am	0.782
14	INESC TECNALIA	Final	INESC TEC Porto / TECNALIA	[link]	Wed, 1 Mar 2017, 7:05:40 pm	0.765
15	Dylan Shen	task3_final_RQ	Computer Vision Institute, Shenzhen University	[link]	Wed, 1 Mar 2017, 9:20:22 pm	0.759
16	Vic Lee	task3_final_Alice	Computer Vision Institute, Shenzhen University	[link]	Wed, 1 Mar 2017, 9:11:31 pm	0.757
17	Masih Mahbod	Skin Lesion Classification Using Hybrid Deep Neural Networks	IPA	[link]	Wed, 1 Mar 2017, 12:51:43 pm	0.715
18	Dennis Murphree	Transfer Learning from Inception	Dennis Murphree	[link]	Wed, 1 Mar 2017, 11:06:33 pm	0.684
19	Hao Chang	MYBrainAI	Yale	[link]	Wed, 1 Mar 2017, 11:53:55 pm	0.636
20	Jaisakthi S.M.	Lesion Classification	SSNMLRG	[link]	Wed, 1 Mar 2017, 9:25:02 pm	0.623
21	Wenhao Zhang	testPhase	CSMedical	[link]	Wed, 1 Mar 2017, 7:08:07 pm	0.500
22	Wiselin Jiji	Dr Jiji P2 Test	Dr Sivanthi Aditanar College of Engineering	[link]	Thu, 2 Mar 2017, 12:46:52 am	0.495
23	Yanzhi Song	submit of yanzhi	song	[link]	Wed, 1 Mar 2017, 8:05:13 am	0.475

Category 2: ROC AUC for Melanocytic Classification

All of the skin lesions that we will examine are caused by abnormal growth of either **melanocytes** or **keratinocytes**, which are two different types of epidermal skin cells. Melanomas and nevi are derived from melanocytes, whereas seborrheic keratoses are derived from keratinocytes.

In the second category, we will test the ability of your CNN to distinguish between melanocytic and keratinocytic skin lesions by calculating the area under the receiver operating characteristic curve (**ROC AUC**) corresponding to this binary classification task.

The top scores in this category (from the ISIC competition) can be found in the image below.

Rank	User	Title	Organization	Documentation	Date	Score
1	monty python	gpm-LSSSD	Multimedia Processing Group - Universidad Carlos III de Madrid	[link]	Wed, 1 Mar 2017, 12:57:35 pm	0.965
2	Kazuhisa Matsunaga	ResNet ensemble with normalized image	Casio and Shinshu University joint team	[link]	Wed, 1 Mar 2017, 11:18:03 pm	0.953
3	RECOD Titans	release (rc36xtrm) "alea jacta est"	RECOD Titans / UNICAMP	[link]	Wed, 1 Mar 2017, 11:42:07 pm	0.943
4	Xulei Yang	multi-task deep learning model for skin lesion segmentation and classification-3	Institute of High Performance Computing + National Skin Center, Singapore	[link]	Tue, 28 Feb 2017, 6:34:10 pm	0.942
5	T D	Last Minute Submission!!!!	University of Guelph - MLRG	[link]	Wed, 1 Mar 2017, 11:55:50 pm	0.935
6	Lei Bi	EResNet (single scale w/o attributes)	USYD-BMIT	[link]	Wed, 1 Mar 2017, 8:04:42 pm	0.921
7	C V	all	icuff	[link]	Tue, 28 Feb 2017, 1:06:44 am	0.911
8	Cristina Vasconcelos	comb	icuff	[link]	Tue, 28 Feb 2017, 1:11:21 am	0.911
9	Masih Mahbod	Skin Lesion Classification Using Hybrid Deep Neural Networks	IPA	[link]	Wed, 1 Mar 2017, 12:51:43 pm	0.908
10	Dylan Shen	task3_final_RQ	Computer Vision Institute, Shenzhen University	[link]	Wed, 1 Mar 2017, 9:20:22 pm	0.886
11	Euljoon Ahn	DeepAhn	USYD-BMIT	[link]	Wed, 1 Mar 2017, 10:30:13 am	0.885
12	INESC TECNALIA	Final	INESC TEC Porto / TECNALIA	[link]	Wed, 1 Mar 2017, 7:05:40 pm	0.881
13	Vic Lee	task3_final_Alice	Computer Vision Institute, Shenzhen University	[link]	Wed, 1 Mar 2017, 9:11:31 pm	0.875
14	Balázs Harangi	Ensemble of deep convolutional neural networks	University of Debrecen	[link]	Wed, 1 Mar 2017, 8:25:16 pm	0.867
15	x j	finalv_L2C1_trir	CVI	[link]	Wed, 1 Mar 2017, 11:17:56 am	0.855
16	Rafael Sousa	Araguaia Medical Vision Lab - GooglAlexNet	Universidade Federal de Mato Grosso	[link]	Wed, 1 Mar 2017, 3:26:22 pm	0.840
17	Matt Berseth	Final Classification Submission	NLPLOGIX / WISEEYE.AI	[link]	Tue, 28 Feb 2017, 6:32:47 am	0.827
18	Dennis Murphree	Transfer Learning from Inception	Dennis Murphree	[link]	Wed, 1 Mar 2017, 11:06:33 pm	0.817
19	Wenhao Zhang	testPhase	CSMedical	[link]	Wed, 1 Mar 2017, 7:08:07 pm	0.817
20	Hao Chang	MYBrainAI	Yale	[link]	Wed, 1 Mar 2017, 11:53:55 pm	0.774
21	Jaisakthi S.M.	Lesion Classification	SSNMLRG	[link]	Wed, 1 Mar 2017, 9:25:02 pm	0.687
22	Wiselin Jiji	Dr Jiji P2 Test	Dr Sivanthi Aditanar College of Engineering	[link]	Thu, 2 Mar 2017, 12:46:52 am	0.498
23	Yanzhi Song	submit of yanzhi	song	[link]	Wed, 1 Mar 2017, 8:05:13 am	0.456

Extracurricular Computer Vision Nangodegr

Category 3: Mean ROC AUC

In the third category, we will take the average of the ROC AUC values from the first two categories.

The top scores in this category (from the ISIC competition) can be found in the image below.

Rank	User	Title	Organization	Documentation	Date	Score
1	Kazuhisa Matsunaga	ResNet ensemble with normalized image	Casio and Shinshu University joint team	[D]	Wed, 1 Mar 2017, 10:18:03 pm	0.911
2	monty python	gpm-LSSD	Multimedia Processing Group - Universidad Carlos III de Madrid	[D]	Wed, 1 Mar 2017, 11:57:35 am	0.910
3	RECOD Titans	release (rc36xtrm) "alea jacta est"	RECOD Titans / UNICAMP	[D]	Wed, 1 Mar 2017, 10:42:07 pm	0.908
4	popleyl .	EResNet (single scale w/o attributes)	USYD-BMIT	[D]	Wed, 1 Mar 2017, 7:04:42 pm	0.896
5	Xulei Yang	multi-task deep learning model for skin lesion segmentation and classification-3	Institute of High Performance Computing + National Skin Center, Singapore	[D]	Tue, 28 Feb 2017, 5:34:10 pm	0.886
6	T D	Last Minute Submission!!!!	University of Guelph - MLRG	[D]	Wed, 1 Mar 2017, 10:55:50 pm	0.886
7	Cristina Vasconcelos	comb	icuff	[D]	Tue, 28 Feb 2017, 12:11:21 am	0.851
8	Cristina Vasconcelos	all	icuff	[D]	Tue, 28 Feb 2017, 12:06:44 am	0.850
9	Euijoon Ahn	DeepAhn	USYD-BMIT	[D]	Wed, 1 Mar 2017, 9:30:13 am	0.836
10	x j	finalv_L2C1_trir	CVI	[D]	Wed, 1 Mar 2017, 10:17:56 am	0.829
11	Balázs Harangi	Ensemble of deep convolutional neural networks	University of Debrecen	[D]	Wed, 1 Mar 2017, 7:25:16 pm	0.825
12	INESC TECNALIA	Final	INESC TEC Porto / TECNALIA	[D]	Wed, 1 Mar 2017, 6:05:40 pm	0.823
13	Rafael Sousa	Araguaia Medical Vision Lab - Goog/AlexNet	Universidade Federal de Mato Grosso	[D]	Wed, 1 Mar 2017, 2:26:22 pm	0.823
14	Dylan Shen	task3_final_RQ	Computer Vision Institute, Shenzhen University	[D]	Wed, 1 Mar 2017, 8:20:22 pm	0.823
15	Vic Lee	task3_final_Alice	Computer Vision Institute, Shenzhen University	[D]	Wed, 1 Mar 2017, 8:11:31 pm	0.816
16	Masih Mahbod	Skin Lesion Classification Using Hybrid Deep Neural Networks	IPA	[D]	Wed, 1 Mar 2017, 11:51:43 am	0.811
17	Matt Berseeth	Final Classification Submission	NLPLOGIX / WISEEYE.AI	[D]	Tue, 28 Feb 2017, 5:32:47 am	0.804
18	Dennis Murphree	Transfer Learning from Inception	Dennis Murphree	[D]	Wed, 1 Mar 2017, 10:06:33 pm	0.750
19	Hao Chang	MYBrainAI	Yale	[D]	Wed, 1 Mar 2017, 10:53:55 pm	0.705
20	Wenhao Zhang	testPhase	CSMedical	[D]	Wed, 1 Mar 2017, 6:08:07 pm	0.658
21	Jaisakthi S.M.	Lesion Classification	SSNMLRG	[D]	Wed, 1 Mar 2017, 8:25:02 pm	0.655
22	Wiselin Jiji	Dr Jiji P2 Test	Dr Sivanthi Aditanar College of Engineering	[D]	Wed, 1 Mar 2017, 11:46:52 pm	0.497
23	Yanzhi Song	submit of yanzhi	song	[D]	Wed, 1 Mar 2017, 7:05:13 am	0.465

Getting your Results

Once you have trained your model, create a CSV file to store your test predictions. Your file should have exactly 600 rows, each corresponding to a different test image, **plus** a header row. You can find an example submission file (sample_submission.csv) in the repository.

Your file should have exactly 3 columns:

- **Id** - the file names of the test images (in the **same** order as the sample submission file)
- **task_1** - the model's predicted probability that the image (at the path in Id) depicts melanoma
- **task_2** - the model's predicted probability that the image (at the path in Id) depicts seborrheic keratosis

Once the CSV file is obtained, you will use the get_results.py file to score your submission. To set up the environment to run this file, you need to create (and activate) an environment with Python 3.5 and a few pip-installable packages:

```
conda create --name derm-ai python=3.5
```

```
source activate derm-ai
```

```
pip install -r requirements.txt
```

Once you have set up the environment, run the following command to see how the sample submission performed:

Extracurricular Computer Vision Nangodegr

`python get_results.py sample_predictions.csv`

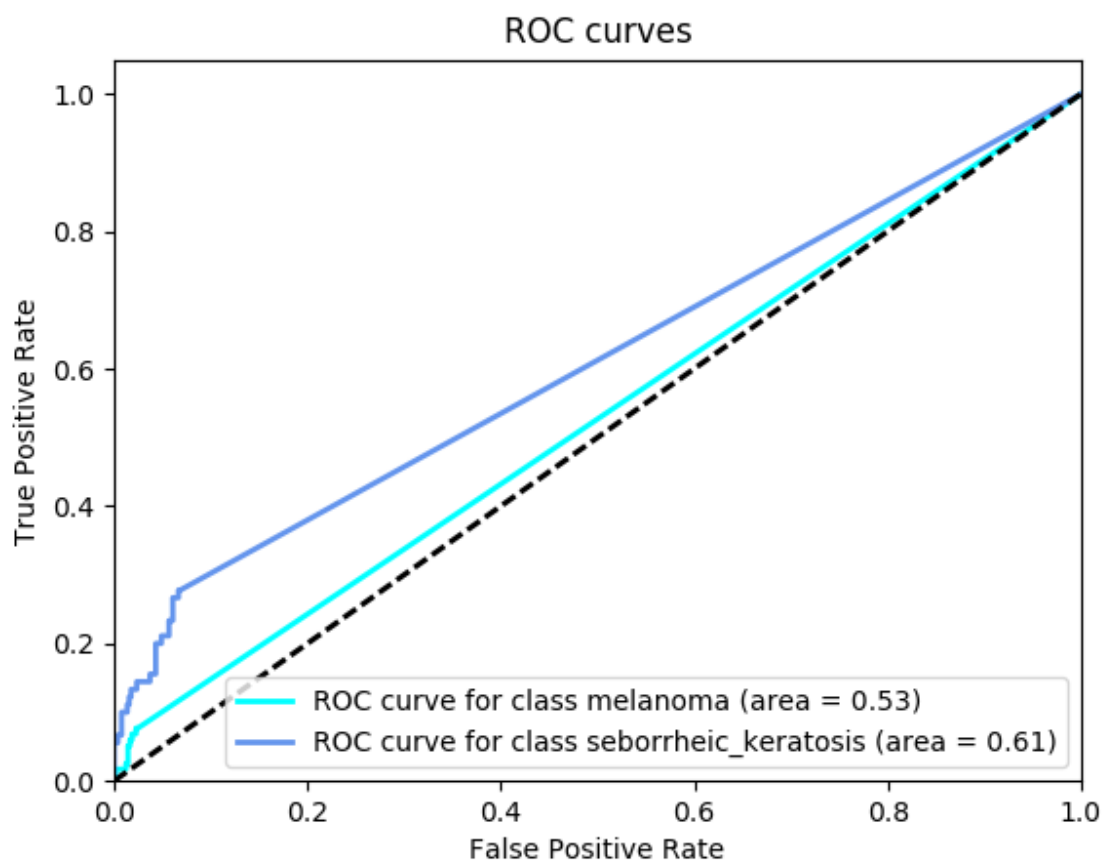
Check the terminal output for the scores obtained in the three categories:

Category 1 Score: 0.526

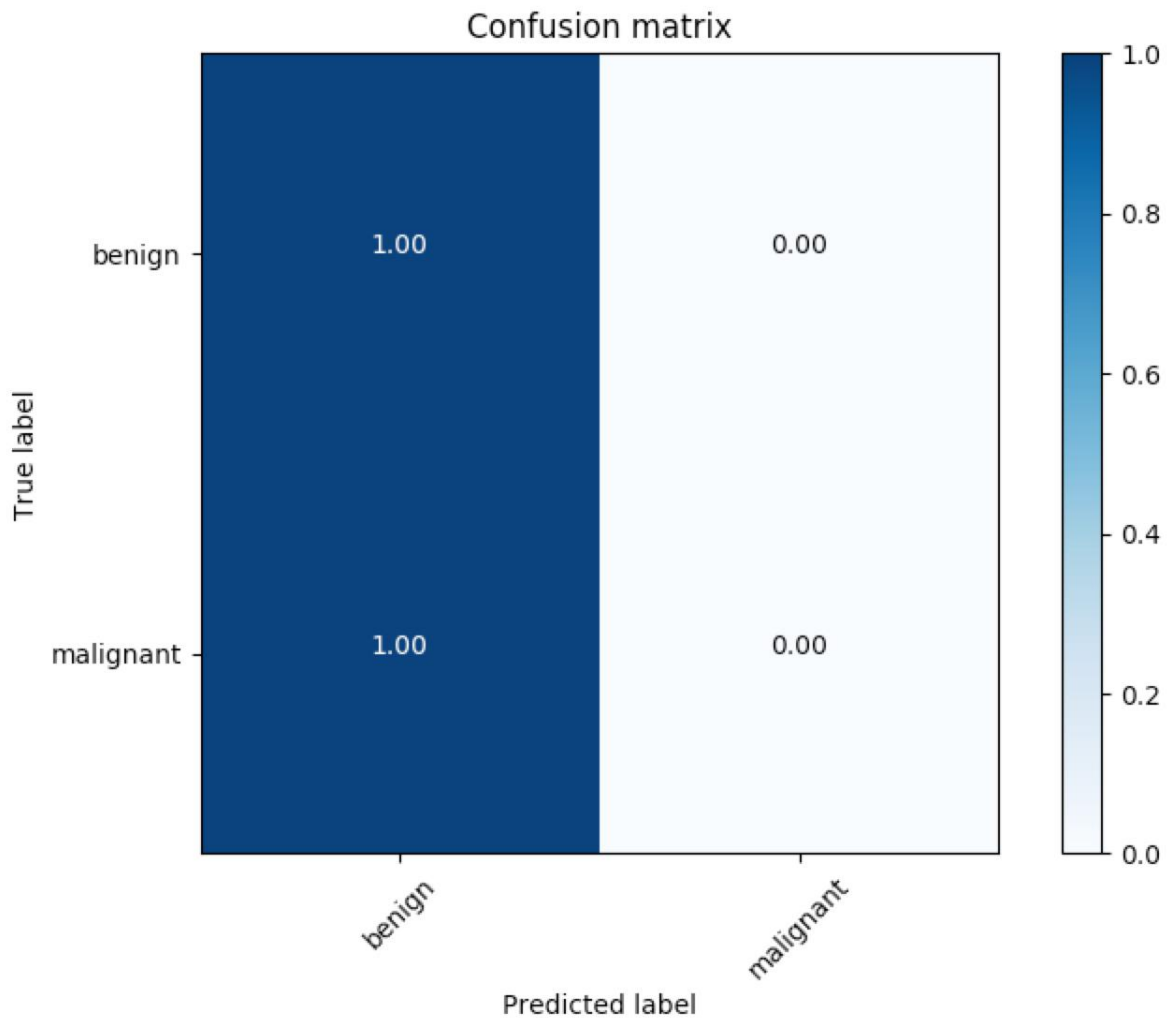
Category 2 Score: 0.606

Category 3 Score: 0.566

The corresponding **ROC curves** appear in a pop-up window, along with the **confusion matrix** corresponding to melanoma classification.



Extracurricular Computer Vision Nanodegree



As you can see from the confusion matrix, the sample submission currently predicts that most of the images in the test dataset correspond to benign lesions. Let's see if your model can improve these results, towards better detecting cancer!

The code for generating the confusion matrix assumes that the threshold for classifying melanoma is set to 0.5. To change this threshold, you need only supply an additional command-line argument when calling the `get_results.py` file. For instance, to set the threshold at 0.4, you need only run:

```
python get_results.py sample_predictions.csv 0.4
```

To test **your own** submission, change the code to instead include the path to **your** CSV file.

Extracurricular Computer Vision Nanodegr

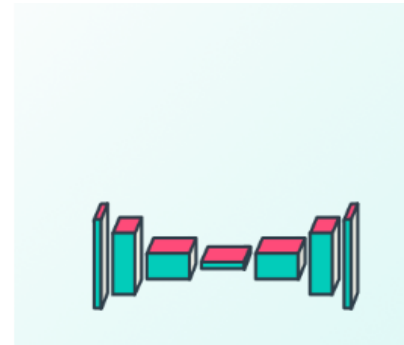
More Deep Learning Models

Lesson 1

LESSON 1

Fully-Convolutional Neural Networks & Semantic Segmentation

Get a high-level overview of how fully-convolutional neural networks work, and see how they can be used to classify every pixel in an image.



https://www.youtube.com/watch?v=1sm1EbfilXI&feature=emb_logo

Why Fully Convolutional Networks (FCNs) ?

https://www.youtube.com/watch?v=WQ_YOz1o9GM&feature=emb_logo

https://www.youtube.com/watch?v=Lh2ozg5yTs&feature=emb_logo

Fully Connected to 1x1 Convolution

https://www.youtube.com/watch?time_continue=6&v=xbPtOhkJW1A&feature=emb_logo

Transposed Convolutions

Transposed Convolutions help in **upsampling** the previous layer to a higher resolution or dimension. Upsampling is a classic signal processing technique which is often **accompanied by interpolation**. The term transposed can be confusing since we typically think of transposing as changing places, such as switching rows and columns of a matrix. In this case when we use the term **transpose**, we mean transfer to a different place or context.

https://www.youtube.com/watch?v=K6mILX8ZZDs&feature=emb_logo

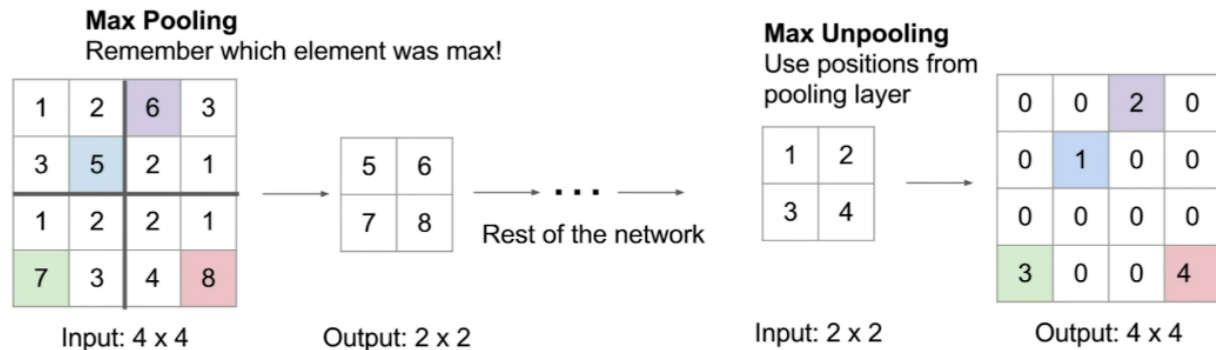
Extracurricular Computer Vision Nangodegr

We can use a transposed convolution to transfer patches of data onto a sparse matrix, then we can fill the sparse area of the matrix based on the transferred information. Helpful animations of convolutional operations, including transposed convolutions, can be found [here](#). As an example, suppose you have a 3x3 input and you wish to upsample that to the desired dimension of 6x6. The process involves multiplying each pixel of your input with a kernel or filter. If this filter was of size 5x5, the output of this operation will be a weighted kernel of size 5x5. This weighted kernel then defines your output layer.

Methods of upsampling

1. Max "unpooling"
2. Learnable upsampling (with transposed convolutions)

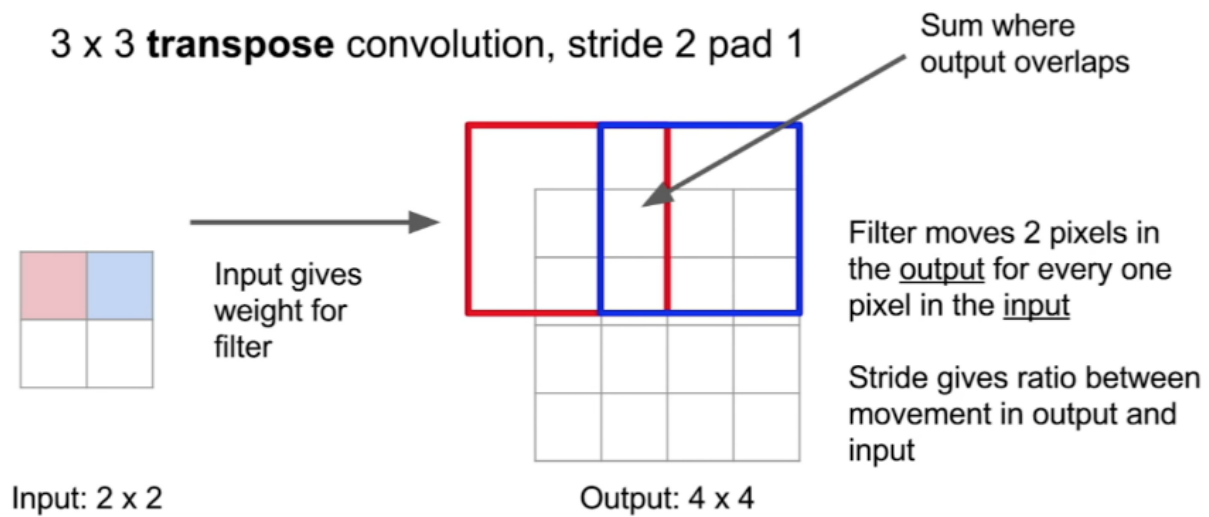
The first is pictured below.



Learnable upsampling, transposed convolutions

We've been going over learnable upsampling layers. The upsampling part of the process is defined by the strides and the padding. Let's look at a simple representation of this. If we have a 2x2 input and a 3x3 kernel and a stride of 2, we can expect an output of dimension 4x4. The following image gives an idea of the process.

Extracurricular Computer Vision Nangodegr



Skip Connections

https://www.youtube.com/watch?v=JUYLA5PWzo0&feature=emb_logo

FCNs In The Wild

https://www.youtube.com/watch?v=q9wTd53-hsw&feature=emb_logo

Bounding Boxes

https://www.youtube.com/watch?v=uPv4d0XI8hc&feature=emb_logo

Semantic Segmentation

https://www.youtube.com/watch?v=L5gJnZrw48&feature=emb_logo

Semantic Segmentation

In semantic segmentation, we want to input an image into a neural network and we want to output a category for **every pixel** in this image. For example, for the below image of a couple of cows, we want to look at every pixel and decide: is that pixel part of a cow, the grass or sky, or some other category?

Extracurricular Computer Vision Nangodegr



Small image of cows in a field.

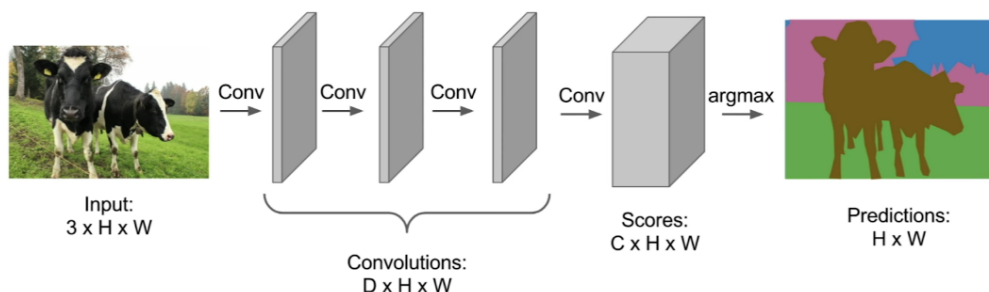
We'll have a discrete set of categories, much like in a classification task. But instead of assigning a single class to an image, we want to assign a class to every pixel in that image. So, how do we approach this task?

Fully-Convolutional Network (FCN) Approach

If your goal is to preserve the spatial information in our original input image, you might think about the simplest solution: simply never discard any of that information; never downsample/maxpool and don't add a fully-connected layer at the end of the network.

We could use a network made entirely of convolutional layers to do this, something called a fully convolutional neural network. **A fully convolutional neural network preserves spatial information.**

This network would take in an image that has true labels attached to each pixel, so every pixel is labeled as grass or cat or sky, and so on. Then we pass that input through a stack of convolutional layers that preserve the spatial size of the input (something like 3x3 kernels with zero padding). Then, the final convolutional layer outputs a tensor that has dimensions $C \times H \times W$, where C is the number of categories we have.



FCN architecture.

Predictions

This output Tensor of predictions contains values that classify every pixel in the image, so if we look at a single pixel in this output, we would see a vector that looks like a classification vector -- with values in it that show the probability of this single pixel being a cat or grass or sky, and so on. We could do this pixel level classification all at once, and then train this network by assigning a loss function to each pixel in the

Extracurricular Computer Vision Nangodegr

image and doing backpropagation as usual. So, if the network makes an error and classifies a single pixel incorrectly, it will go back and adjust the weights in the convolutional layers until that error is reduced.

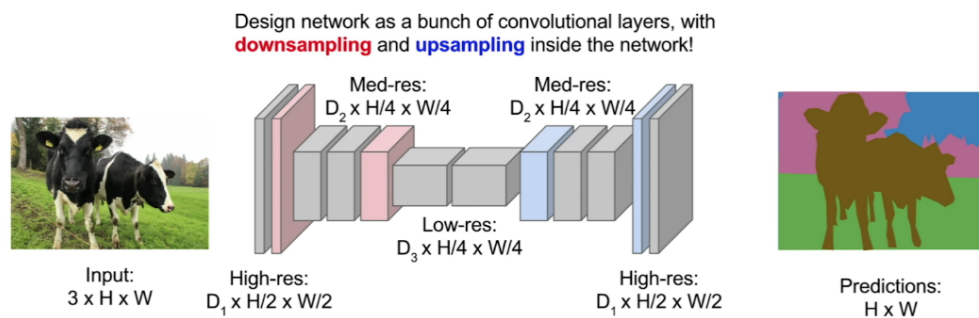
Limitations of This Approach

- It's very expensive to label this data (you have to label every pixel), and
- It's computationally expensive to maintain spatial information in each convolutional layer

So...

Downsampling/Upsampling

Instead, what you usually see is an architecture that uses downsampling of a feature map and an upsampling layer to reduce the dimensionality and, therefore, the computational cost, of moving forward through these layers in the middle of the network.



Downsampling/upsampling on an FCN.

So, what you'll see in these networks is a couple of convolutional layers followed by downsampling done by something like maxpooling, very similar to a simple image classification network. Only, this time, in the second half of the network, we want to increase the spatial resolution, so that our output is the same size as the input image, with a label for every pixel in the original image.

Scene Understanding

https://www.youtube.com/watch?time_continue=7&v=aMQREc-mP50&feature=emb_logo

IoU

Let's walk through an example IOU calculation.

Steps

- count true positives (TP)

Extracurricular Computer Vision Nangodegr

- count false positives (FP)
- count false negatives (FN)
- Intersection = TP
- Union = TP + FP + FN
- IOU = Intersection/Union

True Positive
False Positive
False Negative

	Truth	Predicted		
Class 0	0 0 0 0	1 0 0 0	TP = 3	I = 3
	1 1 1 1	1 3 0 1	FP = 3	U = 7
	2 2 2 2	2 2 2 3	FN = 1	IOU = 3/7
	3 3 3 3	3 1 0 0		
Class 1	0 0 0 0	1 0 0 0	TP = 2	I = 2
	1 1 1 1	1 3 0 1	FP = 2	U = 6
	2 2 2 2	2 2 2 3	FN = 2	IOU = 2/6
	3 3 3 3	3 1 0 0		
Class 2	0 0 0 0	1 0 0 0	TP = 3	I = 3
	1 1 1 1	1 3 0 1	FP = 0	U = 4
	2 2 2 2	2 2 2 3	FN = 1	IOU = 3/4
	3 3 3 3	3 1 0 0		
Class 3	0 0 0 0	1 0 0 0	TP = 1	I = 1
	1 1 1 1	1 3 0 1	FP = 2	U = 6
	2 2 2 2	2 2 2 3	FN = 3	IOU = 1/6
	3 3 3 3	3 1 0 0		

In the above, the left side is our ground truth, while the right side contains our predictions. The highlighted cells on the left side note which class we are looking at for statistics on the right side. The highlights on the

Extracurricular Computer Vision Nangodegr

right side note true positives in a cream color, false positives in orange, and false negatives in yellow (note that all others are true negatives - they are predicted as this individual class, and should not be based on the ground truth).

We'll look at the first class, Class 0, and you can do the same calculations from there for each.

For Class 0, only the top row of the 4x4 matrix should be predicted as zeros. This is a rather simplified version of a real ground truth - in reality, the zeros could be anywhere in the matrix. On the right side, we see 1,0,0,0, meaning the first is a false negative, but the other three are true positives (aka 3 for Intersection as well). From there, we need to find anywhere else where zero was falsely predicted, and we note that happens once on the second row, and twice on the fourth row, for a total of three false positives.

To get the Union, we add up TP (3), FP (3) and FN (1) to get seven. The IOU for this class, therefore, is $3/7$.

If we do this for all the classes and average the IOUs, we get:

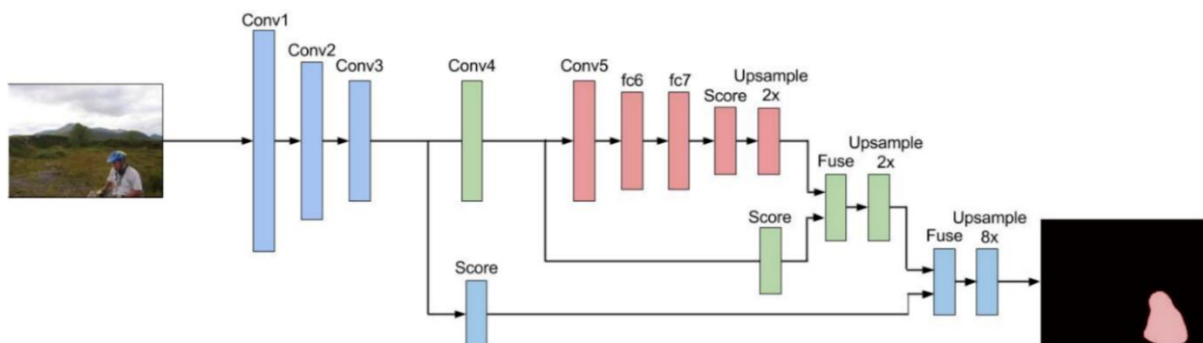
$$\text{Mean IOU} = [(3/7) + (2/6) + (3/4) + (1/6)] / 4 = 0.420$$

FCN-8, Architecture

Let's focus on a concrete implementation of a fully convolutional network. We'll discuss the [FCN-8 architecture](#) developed at Berkeley. In fact, many FCN models are derived from this FCN-8 implementation. The encoder for FCN-8 is the VGG16 model pretrained on ImageNet for classification. The fully-connected "score" layers are replaced by 1-by-1 convolutions. The code for convolutional score layer like looks like:

```
self.score = nn.Conv2d(input_size, num_classes, 1)
```

The complete architecture is pictured below.



There are skip connections and various upsampling layers to keep track of a variety of spatial information. In practice this network is often broken down into the **encoder** portion with parameters from VGG net, and a decoder portion, which includes the upsampling layers.

Extracurricular Computer Vision Nangodegr

FCN-8, Decoder Portion

To build the decoder portion of FCN-8, we'll upsample the input, that comes out of the convolutional layers taken from VGG net, to the original image size. The shape of the tensor after the final convolutional transpose layer will be 4-dimensional: (batch_size, original_height, original_width, num_classes).

To define a transposed convolutional layer for upsampling, we can write the following code, where 3 is the size of the convolving kernel:

```
self.transpose = nn.ConvTranspose2d(input_depth, num_classes, 3, stride=2)
```

The transpose convolutional layers increase the height and width dimensions of the 4D input Tensor. And you can look at the [PyTorch documentation, here](#).

Skip Connections

The final step is adding skip connections to the model. In order to do this we'll combine the output of two layers. The first output is the output of the current layer. The second output is the output of a layer further back in the network, typically a pooling layer.

In the following example we combine the result of the previous layer with the result of the 4th pooling layer through elementwise addition (tf.add).

the shapes of these two layers must be the same to add them

```
input = input + pool_4
```

We can then follow this with a call to our transpose layer.

```
input = self.transpose(input)
```

We can repeat this process, upsampling and creating the necessary skip connections, until the network is complete!

FCN-8, Classification & Loss

The final step is to define a loss. That way, we can approach training a FCN just like we would approach training a normal classification CNN.

In the case of a FCN, the goal is to assign each pixel to the appropriate class. We already happen to know a great loss function for this setup, cross entropy loss!

Remember the output tensor is 4D so before we perform the loss, we have to reshape it to 2D, where each row represents a pixel and each column a class. From here we can just use standard cross entropy loss.

That's it, we now have an idea of how to create an end-to-end model for semantic segmentation. Check out the original paper to read more specifics about FCN-8.

Supporting Materials

[FCN8 Paper](#)

Extracurricular Computer Vision Nanodegree

Text Sentiment Analysis

Lesson 1

Sentiment Analysis

In this lesson, Andrew Trask, the author of *Grokking Deep Learning*, will walk you through using neural networks for sentiment analysis.

CONTINUE →



Materials

As you follow along this lesson, it's extremely important that you open the Jupyter notebook and attempt the exercises. Much of the value in this experience will come from seeing how your solution is different from Andrew's and playing around with the code in your own way. Make this lesson count!

Workspace

The best way to open the notebook is to click [here](#), which will open it in a new window. We recommend you to work on the notebook in that window, and watch the videos in this one. You can also get to the notebook by clicking the "Next" button in the classroom.

If you want to download the notebooks yourself, you can clone them from [our GitHub repository](#). You can either download the repository with git clone <https://github.com/udacity/deep-learning.git>, or download it as an archive file from [this link](#).

This lesson uses the following files:

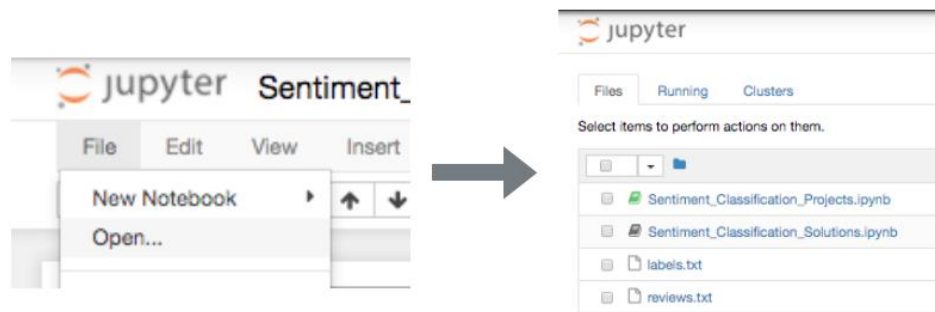
- Sentiment_Classification_Projects.ipynb - a notebook you will use to following along and work on lesson mini projects.
- Sentiment_Classification_Solutions.ipynb - a notebook that includes Andrew's solutions to the lesson projects, which you can use for reference
- A notebook for the solution for each mini project.
- reviews.txt - a collection of 25 thousand movie reviews
- labels.txt - positive/negative sentiment labels for the associated reviews in reviews.txt

Note: the notebooks for these lessons have been updated since the videos were recorded. In most cases that just means your notebook will contain more hints and explanatory text than what you see in the videos, but there may be some minor differences in the code as well. With these changes, you still will be able to follow along with the lessons, and should have an easier time understanding the project material.

Solutions

Extracurricular Computer Vision Nangodegr

If you need help, feel free to look at the solutions in the same folder.



Framing the Problem

https://www.youtube.com/watch?time_continue=6&v=IsTONkAKaJw&feature=emb_logo

Instructions

In this project, you'll test your theory of what features of a review correlate with the label! Here are your specific steps:

Mini Project 1

Task List

Work in the Project 1 section of Sentiment_Classification_Projects.ipynb.

Follow the notebook's instructions to test the correlation between review features and labels.

Solution:

https://www.youtube.com/watch?v=I4r5I0HvHRI&feature=emb_logo

Transforming into numbers:

https://www.youtube.com/watch?v=7rHBU5cbePE&feature=emb_logo

Extracurricular Computer Vision Nangodegr

Mini Project 2

Instructions

In the following mini project, you'll convert the inputs and outputs of the dataset into numbers. Namely, you will convert each review string into a vector, and each label into a `0` or `1`. You'll need to make a few additions to the notebook, but the main work will be implementing two functions, whose signatures are shown below:

```
def update_input_layer(review):
    """ Modify the global layer_0 to represent the vector form of review.
    The element at a given index of layer_0 should represent \
    how many times the given word occurs in the review.
    Args:
        review(string) - the string of the review
    Returns:
        None
    """
    global layer_0
    # clear out previous state, reset the layer to be all 0s
    layer_0 *= 0
    ## Your code here
    pass

def get_target_for_label(label):
    """Convert a label to `0` or `1`.
    Args:
        label(string) - Either "POSITIVE" or "NEGATIVE".
    Returns:
        `0` or `1`.
    """
    pass
```

Extracurricular Computer Vision Nangodegr

- ☐ Work in the `Project 2` section of `Sentiment_Classification_Projects.ipynb`.
- ☐ Follow the notebook's instructions to convert your inputs and outputs to numbers.
- ☐ Create a global vocabulary set `vocab`
- ☐ Initialize a `global` `layer_0` that is a vector of the size of the text vocabulary. All values should be initialized to `0`.
- ☐ Implement `update_input_layer`.
- ☐ Implement `get_target_label`.

https://www.youtube.com/watch?v=45ihpPaeO8E&feature=emb_logo

Building a Neural Network

https://www.youtube.com/watch?time_continue=5&v=aM2k7RTjJI&feature=emb_logo

Mini Project 3

Instructions

In this mini project, you'll modify your old neural network for this specific problem. Here are the specific steps you'll be doing:

Mini Project 3



- ☐ Work in the `Project 3` section of `Sentiment_Classification_Projects.ipynb`.
- ☐ Follow the notebook's instructions to create a neural network that predicts sentiment from movie reviews.
- ☐ Remove the non linearity in your hidden layer.
- ☐ Create training data using the new functions.
- ☐ Create a `preprocess_data` function to create vocabulary for our training.
- ☐ Modify the `train` function to train over the entire corpus.

Extracurricular Computer Vision Nangodegr

Solution:

https://www.youtube.com/watch?v=imnxzCev4SI&feature=emb_logo


Understanding Neural Noise

https://www.youtube.com/watch?v=ubqhh4lv7O4&feature=emb_logo

Mini Project 4

Instructions
In this mini-project, you'll recreate Andrew's work from the previous video to reduce the noise in the neural network. Follow the instructions in the notebook and you should have no problems.

Mini Project 4



☐ Work in the `Project 4` section of `Sentiment_Classification_Projects.ipynb`.

☐ Follow the notebook's instructions and attempt to recreate Andrew's optimizations from the previous video.

Understanding Inefficiencies in our Network

https://www.youtube.com/watch?time_continue=6&v=4MuS-6ATxCU&feature=emb_logo


Extracurricular Computer Vision Nangodegr

Mini Project 5

Instructions

Congratulations on making it all the way to the fifth project! In this mini project, you'll dramatically improve the performance of your network using some of the simple ideas Andrew mentioned. By the end, you'll see how your network can train far faster than before! Here are the specific steps:

Mini Project 5

- ☐ Work in the `Project 5` section of `Sentiment_Classification_Projects.ipynb`
- ☐ Follow the notebook's instructions to speed up forward and back propagation.
-  ☐ Update the neural network that you have to stop multiplying by 0s in the hidden layer.
- ☐ Update the neural network to stop multiplying weights by 1 in the hidden layer.
- ☐ Verify that your neural network has sped up.

https://www.youtube.com/watch?v=Hv86B_jjWTI&feature=emb_logo

Further Noise Reduction

https://www.youtube.com/watch?v=Kl3hWxizKVg&feature=emb_logo


Extracurricular Computer Vision Nangodegr

Mini Project 6

Instructions

You're almost there! In this final mini project, you'll further prune your input data to only use words that have significant signaling power. This will help you improve training time and reduce the size of your training set. Here are your specific instructions:

Mini Project 6

- ☐ Work in the `Project 6` section of `Sentiment_Classification_Projects.ipynb`
- ☐ Follow the notebook's instructions to further reduce noise and speed up training.
-  ☐ Modify `pre_process_data` to take in a `min_count` parameter that specifies the min times a word has appeared.
- ☐ Modify `pre_process_data` to take a `polarity` parameter that determines the absolute value that the polarity of a word must cross.
- ☐ Test your network and see if the training times improve.

https://www.youtube.com/watch?v=ji0famK7gOQ&feature=emb_logo

https://www.youtube.com/watch?v=UHsT35pbpcE&feature=emb_logo

Conclusion:

https://www.youtube.com/watch?v=nIF0GLOQglQ&feature=emb_logo