

In the previous assignment, you implemented and trained your own ConvNets. In this assignment, we will explore many of the ideas we have discussed in lectures. Specifically, you will:

- Reduce overfitting using **dropout** and **data augmentation**
- Combine models into **ensembles** to boost performance
- Use **transfer learning** to adapt a pretrained model to a new dataset
- Use data gradients to visualize **saliency maps** and create **fooling images**

Setup

You can work on the assignment in one of two ways: locally on your own machine, or on a virtual machine through [Terminal](#).

Working locally

Get the code: Download the starter code [here](#).

[Optional] virtual environment: Once you have unzipped the starter code, you might want to create a [virtual environment](#) for the project. If you choose not to use a virtual environment, it is up to you to make sure that all dependencies for the code are installed on your machine. To set up a virtual environment, run the following:

```
cd assignment3
sudo pip install virtualenv      # This may already be installed
virtualenv .env                 # Create a virtual environment
source .env/bin/activate        # Activate the virtual environment
pip install -r requirements.txt # Install dependencies
# Work on the assignment for a while ...
deactivate                     # Exit the virtual environment
```

You can reuse the virtual environment that you created for the first or second assignment, but you will need to run `pip install -r requirements.txt` after activating it to install additional dependencies required by this assignment.

Download data: Once you have the starter code, you will need to download the CIFAR-10 dataset, the TinyImageNet-100-A and TinyImageNet-100-B datasets, and pretrained models for the TinyImageNet-100-A dataset.

Run the following from the assignment3 directory:

NOTE: After downloading and unpacking, the data and pretrained models will take about 900MB of disk space.

```
cd cs231n/datasets
./get_datasets.sh
./get_tiny_imagenet_splits.sh
./get_pretrained_models.sh
```

Compile the Cython extension: Convolutional Neural Networks require a very efficient implementation. We have implemented of the functionality using [Cython](#); you will need to compile the Cython extension before you can run the code. From the `cs231n` directory, run the following command:

```
python setup.py build_ext --inplace
```

Start IPython: After you have downloaded the data and compiled the Cython extensions, you should start the IPython notebook server from the `assignment3` directory. If you are unfamiliar with IPython, you should read our [IPython tutorial](#).

Working on Terminal

We have created a Terminal snapshot that is preconfigured for this assignment; you can [find it here](#). Terminal allows you to work on the assignment from your browser. You can find a tutorial on how to use it [here](#).

Submitting your work:

Whether you work on the assignment locally or using Terminal, once you are done working run the `collectSubmission.sh` script; this will produce a file called `assignment3.zip`. Upload this file to your dropbox on [the coursework](#) page for the course.

Q1: Dropout and Data Augmentation

In the IPython notebook `q1.ipynb` you will implement dropout and several forms of data augmentation. This will allow you to reduce overfitting when training on a small subset of the CIFAR-10 dataset.

Q2: TinyImageNet and Model Ensembles

In the IPython notebook `q2.ipynb` we will introduce the TinyImageNet-100-A dataset. You will try to classify examples from this dataset by hand, and you will combine pretrained models into an ensemble to boost your performance on this dataset.

Q3: Transfer Learning

In the IPython notebook `q3.ipynb` you will implement several forms of transfer learning. You will use adapt a pretrained model for TinyImageNet-100-A to achieve reasonable performance with minimal training time on the TinyImageNet-100-B dataset.

Q4: Visualizing and Breaking ConvNets

In the IPython notebook `q4.ipynb` you will visualize data gradients and you will generate images to fool a pretrained ConvNet.