Project materials

Hello Intro ML students!

This content module includes a set of documents to help you get started on your projects. (Please open them with your NYU Google account.)

For the project component of this course, you will study a state-of-the-art method in machine learning by: learning about your method, replicating some existing version of it (using a Python notebook or source code that is provided for you), and then extending it.

Learn, replicate, and extend? How will I do that?

To learn about your state-of-the-art method, you should expect to read a research paper or other academic material describing the method. You may need to read additional papers or other materials to fill in some between the introductory material in the lectures, and the project topic. For example, if your topic involves a kind of neural network called a transformer, you will probably have to learn about encoders and decoders, attention, and self-attention before you can begin to understand the transformer. We'll share resources on Piazza (and hopefully your classmates will, too!) to help you get up to speed.

To replicate the work, you'll use a Python notebook or open-source Python code that will be given to you. Depending on the specific topic, you may find that the notebook provided to you runs on Colab without any changes necessary. Other students may have to do some more work to: retrieve data into the Colab workspace, install Python libraries or specific versions of Python libraries, make small changes to the code to work with different library versions, or similar modifications.

Finally, you'll extend the replication. You'll have to decide the best way to extend the work, but it's important that you do something that shows your understanding of the method. Some examples:

- If the paper that describes your method discusses some of its strengths and weaknesses, you might use specially prepared test samples to show how the model does well on test samples that fall within its "strengths" and poorly on test samples that fall within its "weaknesses".
- You might use a completely different type of test data, explain why you would expect the model to perform well (or not!) on that type of data, and show what you could do to improve the model performance on that type of data.
- You might compare and contrast different methods for a similar problem using the same data, and explain any differences in their performance using your understanding of the methods.

I have a topic, now what?

Your first step will be to select a topic! Here is an overview of the timeline:

Date	Task	Expected effort (hours)
October 21-31	Students choose a project partner and sign up for a project from a list.	1
November 1-30	Students work with their partner to replicate the existing work, extend it, prepare their Colab notebook for submission, and prepare a brief presentation.	~20
December 1- 14	Students submit their notebook and present an overview of the work to instructor and other students.	2

Next steps

Once you have confirmed your project topic, and when you're ready to start working on your project, I recommend the following:

1. Start by getting some baseline code running (on Colab)

All of the projects include a link to some code - one or more notebooks, Github repositories, blog posts with code snippets included, or other source code that you can use as a baseline for your project.

A good first step is to get some baseline code running. I shared tips and solutions to common problems in another document!
Some of you have a notebook that will run as-is without any changes, so you don't have to do anything to get your baseline code running. Some of you will need to make a few small changes, like changing the Tensorflow version from 2.x to 1.x, or installing some packages, in order to get your baseline code to run. Some of you will need to make more substantial changes.

Why should this be your first step? If you can't get your baseline code working, you should reach out to me for help as soon as possible, so that I can work with you to get something running. If we can't get your baseline code working together, then I'll help you get started on a different, related project instead.

If you spend a lot of time and effort on your project topic *before* you get some code running, that time and effort might be wasted if you end up having to switch topics.

2. Learn about your project topic

Once you have your baseline code running, a good next step is to do a deep dive into your project topic and learn much more about it. Read the paper where the topic is introduced, and any other materials (blog posts, etc.) that I recommended. You should also seek out additional high-quality learning resources on your own (make sure to keep track of all your sources, so that you can cite them properly in your report!)

Look for details like:

- How exactly does this technique work?
- What other techniques are there for similar problems? What is special about this technique, compared to the others?
- What kind of problems/data would this technique be best suited for? What kind of problems/data would it not perform well on? Why?
- Are there other techniques that perform better than this one under some circumstances? If so, is there any circumstance in which this technique is still the best?

You can use your baseline code to help you understand the topic, too. Look at the source code, and try to understand how it connects to the details you've read about how the technique works. You may add extra visualizations or other output to your baseline code to help you understand the technique.

3. Plan to extend your work

After you have some baseline code running *and* you have a good grasp of the topic/technique, you are in a good position to extend the baseline in a way that shows your understanding of the topic.

A good original contribution is one that shows that you understand the technique, how it is used, and what its strengths and weaknesses are. For example, you might apply it to a problem it is especially well suited for (and explain why it is!), compare it to another technique that is used for similar tasks (and make sure to explain the results!), show how it works with some carefully selected test samples that are designed to highlight its strengths and weaknesses, etc.

If you're not sure about your planned extension, feel free to ask for feedback!

Getting help

This project asks you to go a few steps farther than what we have covered in class. It's supposed to be challenging, but if you get stuck, you don't have to struggle alone! Here is how to get help.

You can get help with conceptual questions (Example: "I can't figure out what attention layers do. All of the online resources I can find - I listed them below - are about attention layers for NLP problems, but my problem is about understanding images, and I don't know what attention layers do in that context.")

You can also get help with technical questions (Example: "I know I need to install some specific package versions to get my notebook to run, but I can't figure out which! These are all the things I have tried...")

And, you can get help with planning how to extend the existing work (Example: "My baseline code does pose estimation on a YouTube video. Here's a link to the notebook. Would running this on a different YouTube video be a good extension?")

To get help, you can:

- Ask on Ed! Feel free to include a link to your draft Colab notebook, if relevant, but make sure to adjust the sharing permissions so that I can see it.
- Attend office hours. You can find the Zoom link in the calendar in Brightspace.

Presentation guidelines

As we wrap up our project work, you should be starting to prepare your presentations! First, some quick notes on logistics:

- Each team will prepare a presentation (with slides, etc.) and use it to deliver a 7-**minute** talk on their project. Yes, that's a very short presentation!
- The time limit is a **strict** time limit. You should not exceed 7 minutes.

- Your presentation should give a high-level overview of the topic (~3 minutes), explain and show the results of your replication of the existing code (~1 minute), and also describe how you extended the existing code (and show results) (~2 minute).
- Even though your own presentation is only 7 minutes, you will attend an entire presentation time slot (which may be 1 or 2 hours long, depending on the timing) and listen to your peers' presentations.
- You can pre-record your presentation, or you can deliver it live (your choice). If you pre-record, you will have to send me the recording in advance, and you still have to attend the live session to answer questions and interact with the audience.
- After your presentation, I will ask you a couple of questions about your project.

Next, some comments on what I'll be looking for when grading the presentations. Pay careful attention to these guidelines!

- A very brief presentation is a real test of how well you understand the topic! Anyone can give a long presentation in which they repeat lots of technical detail that they don't really understand; only someone who understands a topic very well can *effectively* synthesize the most important parts and convey it well in just a few minutes.
- A presentation that is technically correct is a necessary but not sufficient condition to get a very good grade. To get a very good grade, it should be technically correct AND also show excellent understanding of the topic, by choosing the most appropriate details to include and explaining them well.
- The technical level of the presentation should be appropriate for Intro ML students who are *not* experts in your specific project topic. In other words, present to your peers! Assume your audience has some knowledge of ML basics (you don't have to explain what a neural network is), but does not have all the specific knowledge that you acquired while working on your project (you can't make casual references to "Thompson Sampling" or "Zero-Shot Classification" without explaining what you mean!). Try to avoid using lots of unnecessary jargon.
- A presentation is a visual medium; use it effectively. Avoid slides with lots of text. Use images that you'll refer to as you speak, to help illustrate key points. (But, do not include images that you don't actually use or refer to: if you show an image in a slide, I expect that you're going to explain what I'm supposed to understand from it.)
- Attribution is **very important**:
    - At the beginning of your presentation, include a citation to the paper that your project is based on.

    - Throughout your presentation, whenever you use images that you didn't create yourself, cite the source of the image (in a small caption right near the image).
    - Similarly, if you use text or ideas from somewhere else, cite the source. For example, if you have a slide explaining the benefits and disadvantages of different instance segmentation models, and you got these from a post on Reddit - include a small note on the bottom of the slide citing the source. If you got the idea for your extension of the project from a Medium post, cite it in a small note on the bottom of the slide. Etc.
- Your overview of the topic should explain what problem this technique is designed to solve, and give the intuition behind the technique: how does it work, what makes it important, what makes it better or worse than other techniques, etc.
- Your discussion of the replication and extension should mention the most important details of what you did (did you train a model or use a pre-trained model? what data did you use? etc.), but you won't have time to show a lot of detail about the code.
- Make sure to explain why you chose to extend the original work the way you did. How does your extension show us something interesting, and that is *specific* to this technique? (For example, an extension that just trains a model with different learning rates doesn't show anything interesting about that specific model. This wouldn't be a good way to "extend" a project.)
- I may ask questions about anything *you* mention during your presentation. If there's something you don't really understand, you're better of not mentioning it at all. Focus on what you understand best and can explain well.