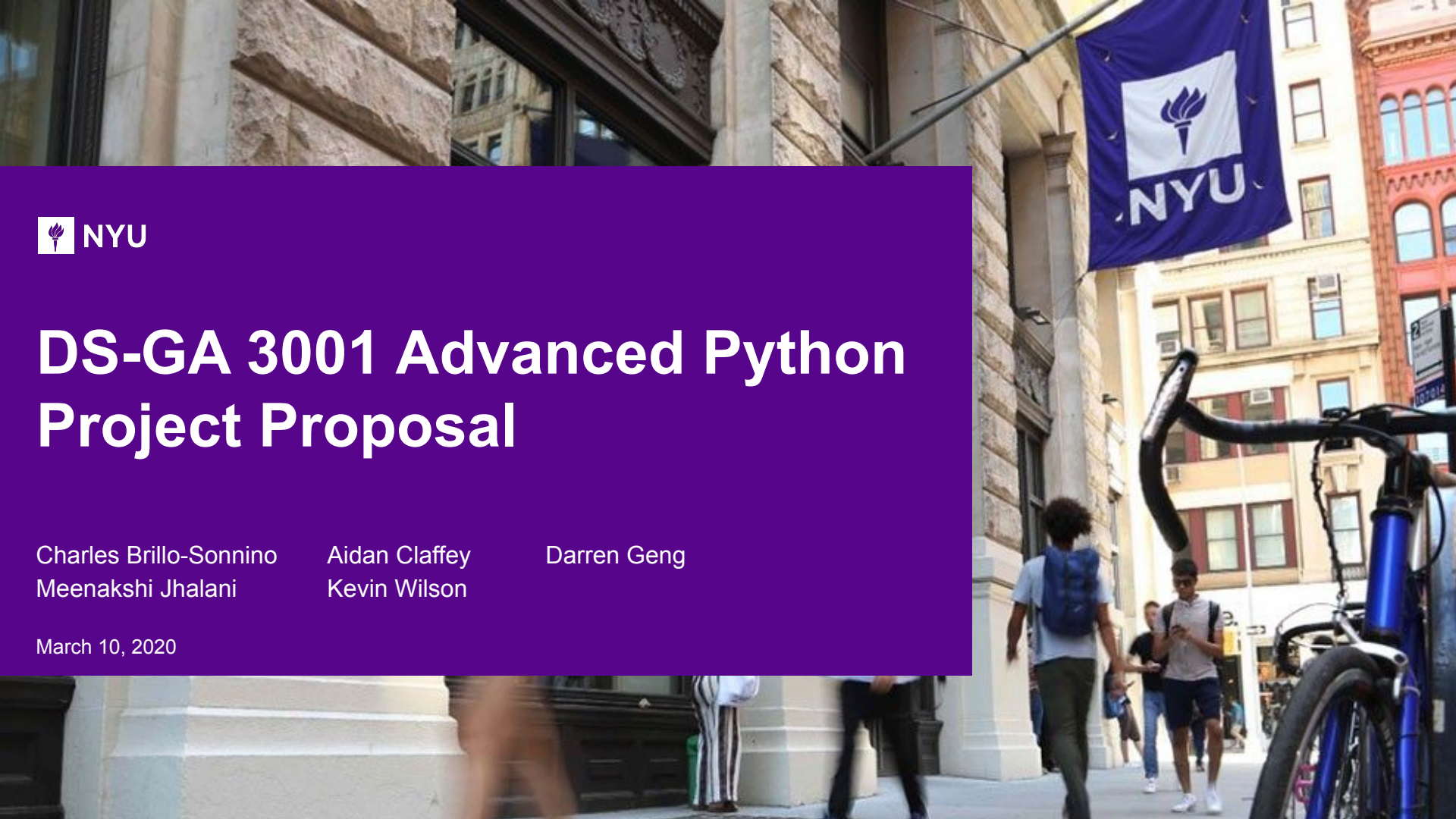


NYU

DS-GA 3001 Advanced Python Project Proposal

Charles Brillo-Sonnino
Aidan Claffey
Darren Geng
Meenakshi Jhalani
Kevin Wilson

March 10, 2020

NYU

Agenda

- Project Overview
- Dataset
- Machine Learning Model Architecture
- Advanced Python Tools We Will Implement

*

IMAGE CAPTION GENERATOR

How would you caption this?

Two Components:

- Computer Vision - interpret image
- Natural Language Processing - converting image to words

Why?

- Describe images to people who are visually impaired
- Describe video in real time

Dataset: “FLICKR 30K”

- Dataset of 31,783 images (.jpg files totaling 4.5GB) obtained from Flickr, mainly of people involved in everyday activities and events

- 158,915 crowd-sourced captions (5 per image)
  - Multiple captions for each image are taken because there is a great amount of variance that is possible in the captions that can be written to describe a single image
  - Also satisfies the dynamic nature of images

http://shannon.cs.illinois.edu/DenotationGraph/data/flickr30k.html

Dataset: “FLICKR 30K”

Example image and associated captions

“An elderly man with light brown hair, wearing a gray sweater, blue shirt, brown pants and tan shoes, reads a book sitting at a bench, as two ducks feed themselves on the grass.”

“A man with parted hair and wearing glasses is seated outdoors on a bench where he is reading.”

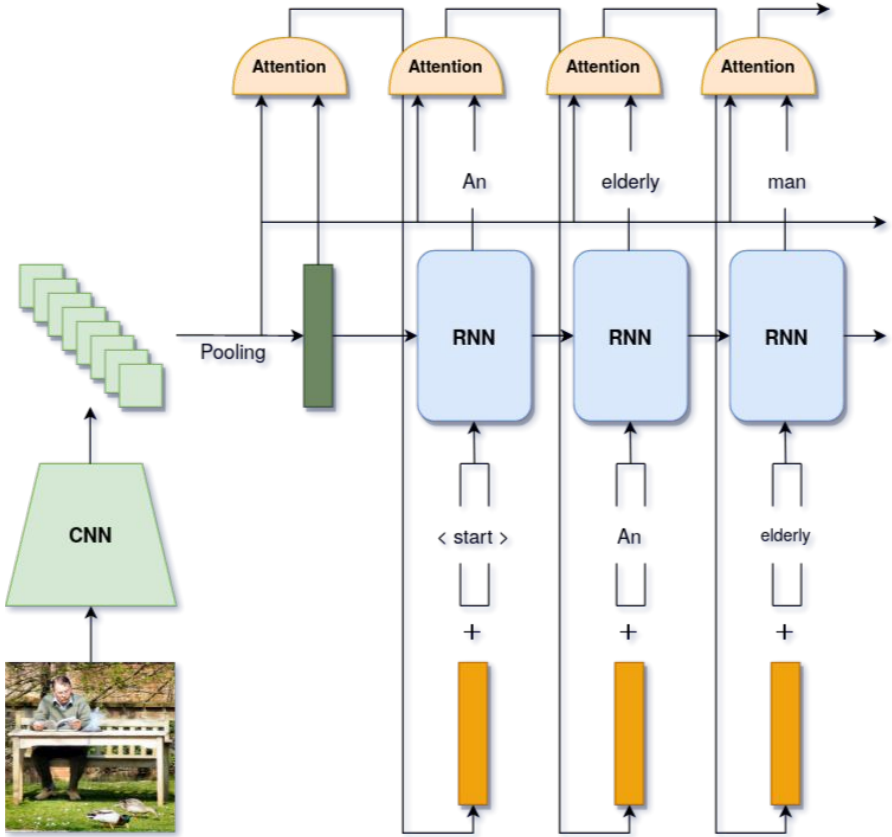
“A man sits outside at a wooden table and reads a book while ducks eat in the foreground.”

“An elderly man sitting on a bench while reading a book.”

“Man reads in a park while feeding the ducks.”

Model Architecture

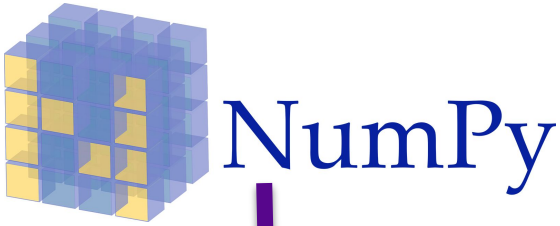
- **Encoder-Decoder**
  - **Encoder** - CNN to encode image into a learned “feature” representation
  - **Decoder** - RNN with self-attention to sequentially generate sentence from Encoder’s output
- Experiment with pre-trained and self-trained encoder
- BLEU score (or similar metric) to calculate validation performance



Use of “Advanced Python” Concepts

Pre-processing

- Efficient mean & standard deviation computation of images for normalizing RGB channels (NumPy, Numba)
- Efficient word tokenization (spaCy, written in Cython)
- Performance tuning through line_profiler



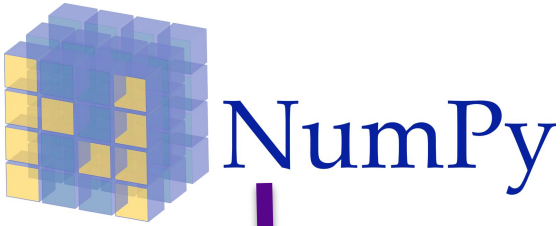
Use of “Advanced Python” Concepts

Training

- GPU acceleration (numba.cuda, week 12 techniques)
- Efficient pipelining (using line_profiler to look for bottlenecks)

Prediction

- Beam Search (with n-gram blocking)
  - Algorithmic optimization
  - Parallel computation
  - Cython and/or Numba compiling to improve computation speed



NYU

Questions?