sdmay19-10: Distributed Wild Bird Surveillance and Recognition System

Week 7 Report
October 22 - October 29

Team Members

Claudia Athens — *Systems Integration Lead*
Ben Simon — *User Interaction Lead*
Francisco Arreola — *Infrastructure Lead*
Pierce Adajar — *Machine Learning Lead*

Client

Dr. Joseph Zambreno

Faculty Advisor

Craig Rupp

Summary of Progress this Report

This week we worked on creating a prototype demo of our project for the technical challenge lightning talks. The demo entailed developing our image storage and retrieval, setting up our camera system, capturing/storing images, and reworking our classification system. The image system is being stored on a Google Cloud server system. Here, images from the Jetson board are being saved the cloud. We have demonstrated that we can store and retrieve images from this website using our Python server.

Past Week's Accomplishments

- Board has been flashed with E-Con Systems camera drivers
- Camera has been integrated with the board
- OpenCV is downloaded and built for the TX1
- We can now save images, and display (but not save) video
- Progress has been made toward beginning to train a modified SqueezeNet architecture. The architecture has been worked out (see below)
- Input pipeline has been constructed
- Displaying images from the board to the cloud to the web interface

---

Pending Issues

A pending issue we are having is the ownership of the cloud project. Francisco currently has his credit card attached to the Google Cloud. We hope to have a seperate account to pay for the Google Cloud account for his own security. Also, the image capture program we created that just uses the OpenCV functions to compress and image to a .png file takes approximately ten seconds to run. We will have to somehow reduce this time because that will take too long for a real time application.

Plans for Upcoming Reporting Period

This week we will work on refining the data pipeline, object detection, video streaming, and creating a more refined front end.

Individual Plans:

Ben:

- Continue to learn Flask and how to use Jinja to serve web pages
- Create a clean web interface to display images using jinja and AngularJS

Claudia:

- Install CUDA, CUDNN, and YOLO on the board
  - Run YOLO on captured images
- Try to get captured videos saved
- Write OpenCV program to detect objects to automate the image capture progress
- Try to reduce time it takes to run camera capture program

Francisco:

- Improve picture uploading script
- Begin work on streaming video from the jetson to the cloud

Pierce:

- Scripts to save / load tf.Dataset object - will speed up training iterations
- Implement worked out model architecture - see attached images for details
- Stretch goal: implement training monitoring hooks

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Claudia Athens	Flashed the board with the camera's drivers. Installed OpenCV and other necessary programs on the board. Wrote OpenCV application that can capture/save an image and take a video.	17	68.5
Ben Simon	This week I worked create the web interface for the demo. The interface downloads and displays all images from the google cloud on a web page. Secondly, I helped installed ubuntu to reflash the jetson.	15.5	67.5
Francisco Arreola	Configured cloud storage project and wrote a variety of python scripts to upload and get images from the cloud storage. Aided ben in website development and working with GCP storage. Created bash script to automatically upload pictures from the camera to the cloud.	16.5	68
Pierce Adajar	Aided in flashing the new drivers to the board. Constructed input pipeline - primarily because the second version of the birds dataset will not fit in my RAM; this allows us to use the newer version of the dataset. Worked out the architecture modifications.	16	67.5

Meeting Notes With Client

No meeting with our client.

Gitlab Activity Summary

- Oct 24th - Pierce worked on the rework ml model. Created data pipeline
    - This commit replaces the previous functionality of create_dataset:
      - Before, you would run create_dataset and load the stored values after execution
      - Now, you can call the functions defined in create_dataset directly.
      The function create_dataset() will return a tf.data.Dataset object, which
      is already set up for you with the given batch_size.
      Note that this design paradigm offloads the burden of creating an appropriate
      tf.data.Iterator to the caller - this was not necessary previously since
      the dataset was returned as a pair of lists rather than a proper Dataset.
- Oct 26th - Francisco added code for a generic python Flask web service and experimented with the google cloud service.
- Oct 27th
    - Francisco worked on google cloud server configuration
- Oct 28th
    - Ben worked on creating a test Flask service to serve a webpage
- Oct 29th
    - Francisco worked on creating a Bash script to send pictures off the board
    - Ben worked on developing the test server and created a web page that server images from google cloud

---

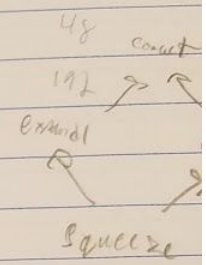
	input	120 × 120 × 3	120×120×3
	conv 1	7×7 stride 2 (96 filters)	57×57×96
	maxpool 1	3×3 stride 2	28×28×96
fire 2	squeeze 2	1×1 s 1 × 16	28 × 28 × 16
	expand 2 →	1×1 s 1 × 64	28 × 28 × 64
	→	3×3 s 1 × 64 pl (pad 1)	28 × 28 × 64
	concat		28 × 28 × 128
fire 3	squeeze 3	1×1 s 1 × 16	28 × 28 × 16
	expand 3 →	1×1 s 1 × 64	28 × 28 × 64
	→	3×3 s 1 × 64 pl	28 × 28 × 64
	concat		28 × 28 × 128
fire 4	squeeze 4	1×1 s 1 × 32	28 × " 32
	expand 4 →	1×1 s 1 × 128	" " 128
	→	3×3 s 1 × 128 pl	" " 128
	concat		" " 256
	maxpool 5	3×3 s 2	13 × 13 × 256
fire 6	squeeze 6	1×1 s 1 × 32	13 × 13 × 32
	expand 6 →	1×1 s 1 × 128	" " 128
	→	3×3 s 1 × 128 pl	" " 128
	concat		" " 256
fire 7	squeeze 7	1×1 s 1 × 48	" " 48
	expand 7 →	1×1 s 1 × 192	" " 192
	→	3×3 s 1 × 192 pl	" " 192
	concat		" " 384
fire 8	squeeze 8	1×1 s 1 × 64	" " 64
	expand 8 →	1×1 s 1 × 256	" " 256
	→	3×3 s 1 × 256 pl	" " 256
	concat		" " 512
	maxpool 9	3×3 s 2	7 × 7 × 512
10	squeeze 10	1×1 s 1 × 64	" " 64
	expand 10 →	1×1 s 1 × 256	" " 256
	→	3×3 s 1 pl × 256	" " 256
			" " 512
	concat		" " 200
	conv 11	1×1 s 1 × 200	1 × 1 × 200 → output
	avgpool 12	7 × 7 s 1	

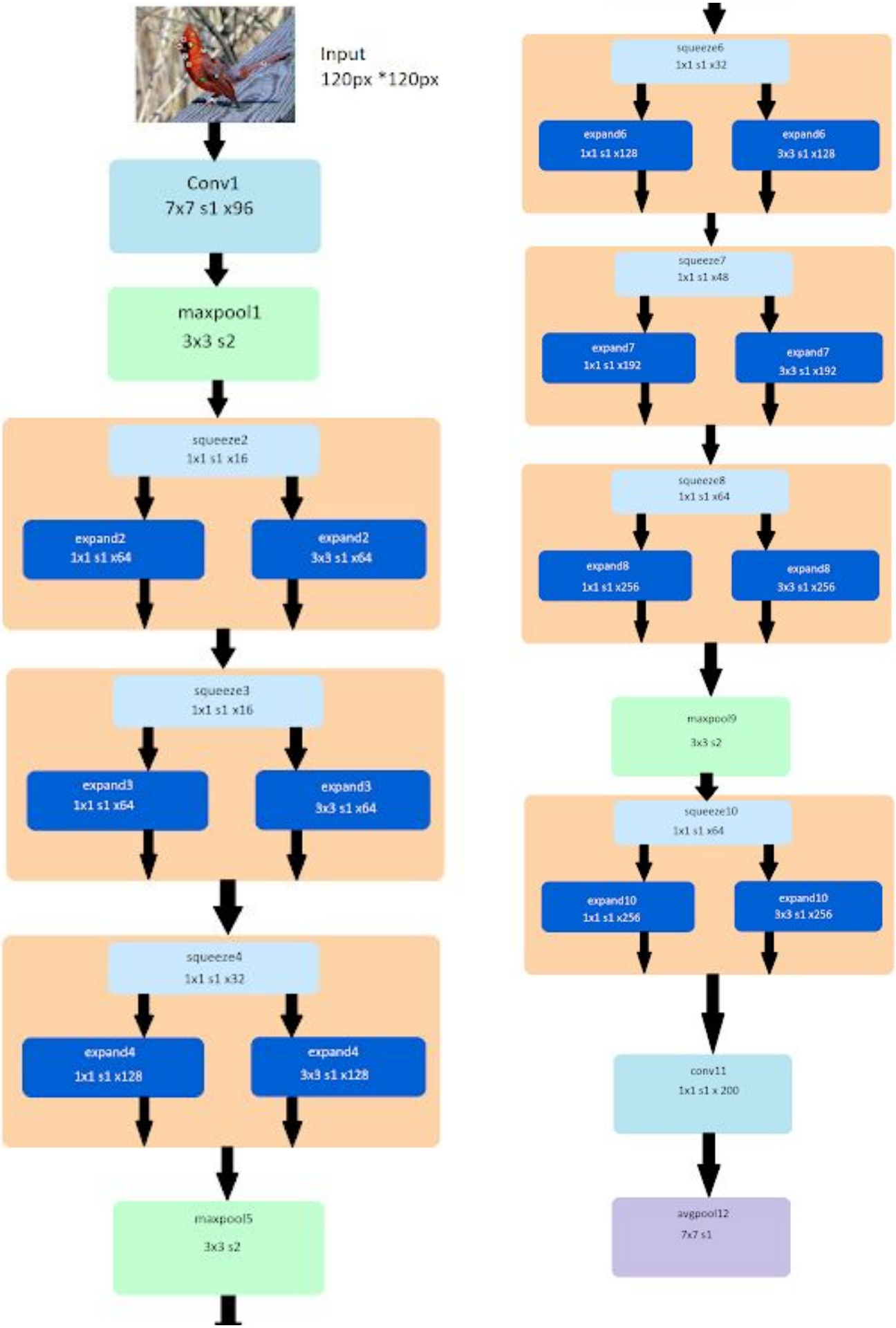
d = dimension
F = filter
S = stride
P = padding

valid: $\text{conv}\left(\frac{d - F_d + P}{S_d}\right)$

⊗ Pre-pad 3×3 with 1-borders

"FC" layers




Input
120px *120px
Conv1
7x7 s1 x96
maxpool1
3x3 s2
squeeze2
1x1 s1 x16
expand2
1x1 s1 x64
expand2
3x3 s1 x64
squeeze3
1x1 s1 x16
expand3
1x1 s1 x64
expand3
3x3 s1 x64
squeeze4
1x1 s1 x32
expand4
1x1 s1 x128
expand4
3x3 s1 x128
maxpool5
3x3 s2
squeeze6
1x1 s1 x32
expand6
1x1 s1 x128
expand6
3x3 s1 x128
squeeze7
1x1 s1 x48
expand7
1x1 s1 x192
expand7
3x3 s1 x192
squeeze8
1x1 s1 x64
expand8
1x1 s1 x256
expand8
3x3 s1 x256
maxpool9
3x3 s2
squeeze10
1x1 s1 x64
expand10
1x1 s1 x256
expand10
3x3 s1 x256
conv11
1x1 s1 x 200
avgpool12
7x7 s1