Winter Progress Report

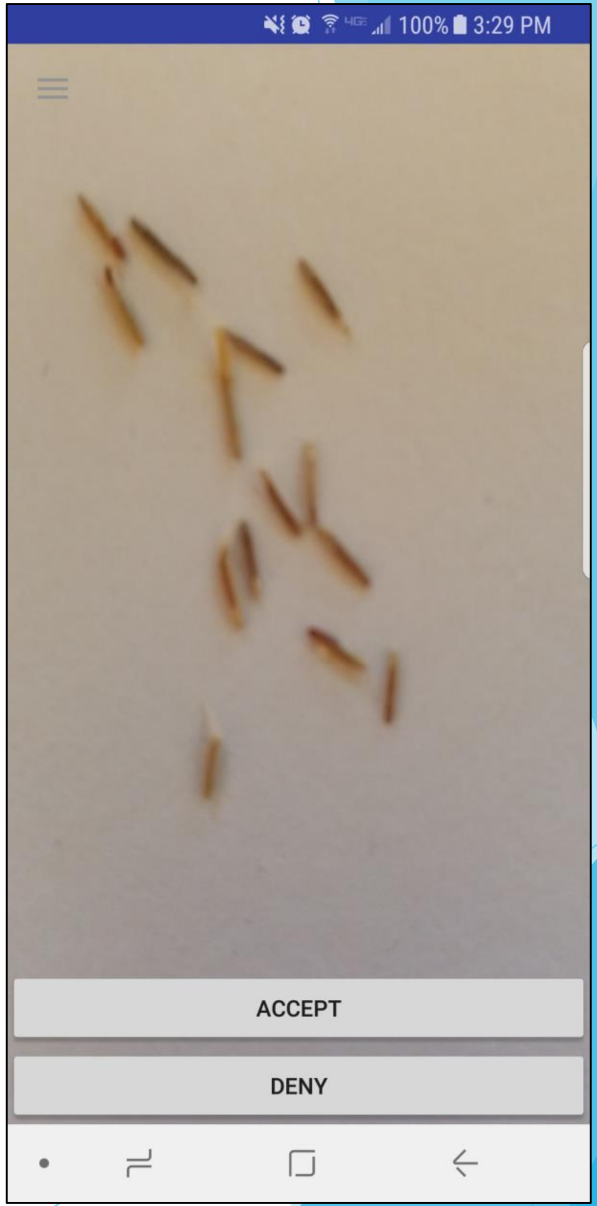
Alexander Ruef, Ethan Takla, Quanah Green

Overview

- State of the project
- Auto-annotation tool demonstration
- Real-time sample analysis with smartphone
- Conclusion

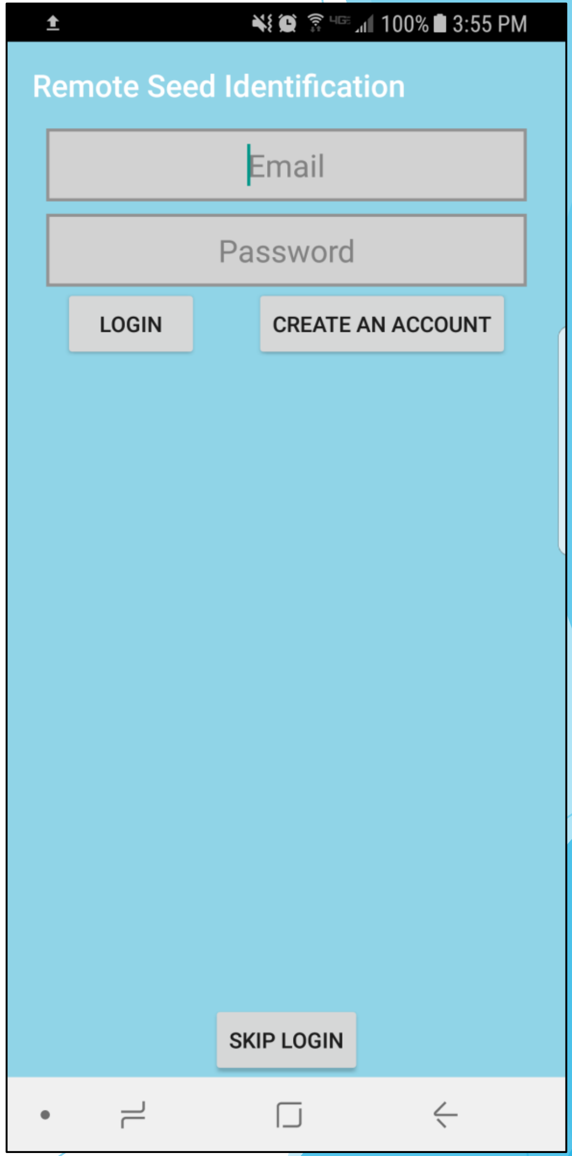
Android Application: Progress

- Server connection established
- Camera features added
- UI groundwork set



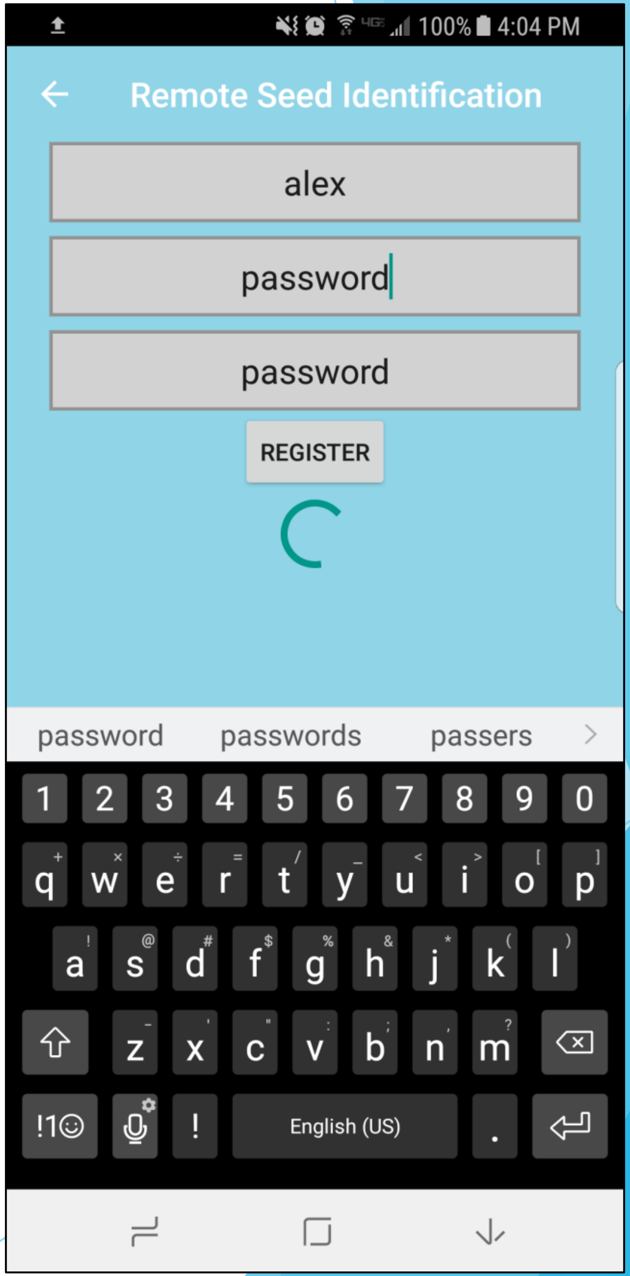
Android Application: To do

- Server error handling
- Camera features
- UI improvements



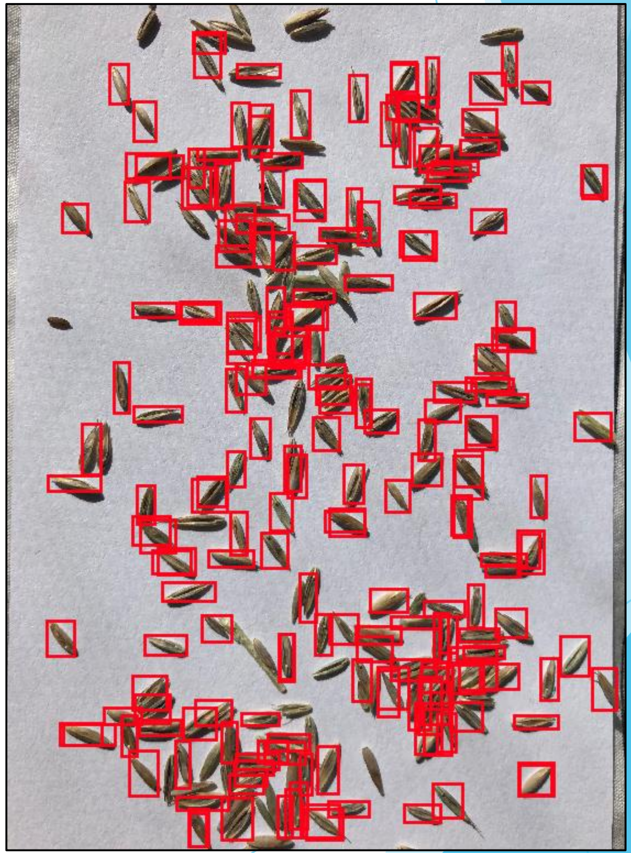
Android Application: Issues

- Establishing socket connection
- Handling callbacks from server
- Preparing and sending images



Classifier: Progress

- Moved to higher resolution framework for more accuracy
- 20,000+ sample photos taken
- 40 cumulative hours of annotating reached
- Full-sized sample image classification functional with up to 500 seeds in an image
- Auto-annotation tool complete



Classifier: Issues

- Database needs to be re-built after switching to higher-resolution framework
- Auto-annotator trained on a small dataset (~800 samples) doesn't do a good very good job at predicting bounding boxes
- Auto-annotator needs to be trained on ~3000 images from each species to be accurate. Once we have hand-annotated these images, annotation will become much easier

Classifier: To do

- Implement bounding box merging for a cleaner, more accurate result
- Finish 20,000 image database and train
- Implement proper pixel-mean shifting for improved training performance
- Stretch goal: Use ResNet instean of VGG-16

Back-end: Server

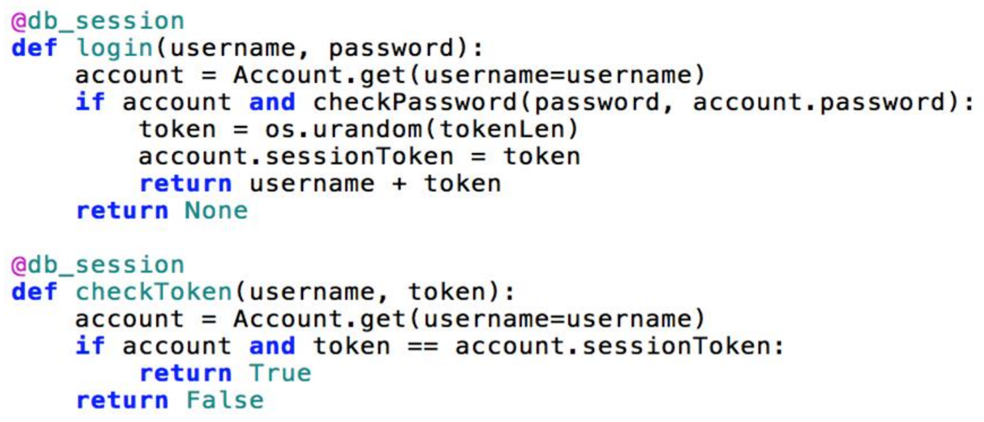
- Upgraded from a normal socket server to a TLS/SSL encrypted socket server using pythons native SSL module.
  - Using a self signed ssl certificate and hosting the server on my home network
  - Need to figure out where and how to host on campus and get a signed ssl certificate
- Created and added a cookie-like login token model to avoid repeated password validation.
- Still working with a single-threaded model, but will switch to a multi-threaded model soon.

Back-end: Protocol

- Modified protocol to include message length, allowing multiple messages to be sent without closing the connection.
- There are 6 different types of server request, differentiated by a byte flag in the message
  - Create Account
  - Login
  - Request Analysis
  - Request a list of reports generated by a user
  - Request a specific report
  - Logout

Back-end: Database

- Still using PonyORM with SQLite
- Now hashing and salting passwords before storage, so they are no longer stored as plain text
- Using pbkdf2 with 100,000 iterations of sha512 and a random 16 byte salt
- Made modifications to support login tokens



```
@db_session
def login(username, password):
    account = Account.get(username=username)
    if account and checkPassword(password, account.password):
        token = os.urandom(tokenLen)
        account.sessionToken = token
        return username + token
    return None

@db_session
def checkToken(username, token):
    account = Account.get(username=username)
    if account and token == account.sessionToken:
        return True
    return False
```

Back-end: Jetson

- Got the Jetson up and running
- Gave it a DNS address so everyone on the team can access it via ssh
- Installed and tested server code
- Got the classifier running on the Jetson
  - Had to install a lot of dependencies, some of which didn't explicitly support the architecture
  - Had to compile some python modules from source
- Integrated the classifier and the server code
  - No more placeholders!
- Analysis takes about 5 minutes

Demo: Automated seed annotation

Demo: Real time sample analysis with Android app

Conclusion

- Fully functional sample analysis on a smart-phone achieved
- Automated seed annotation speeds up dataset generation
- Back-end upgraded with security and Jetson processor
- Still need to build final dataset