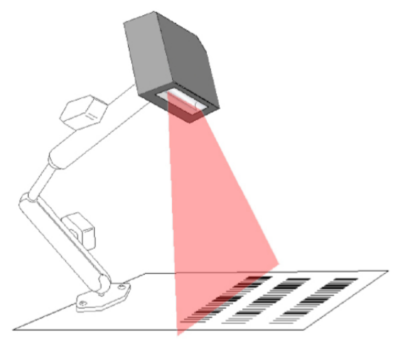
Computer Vision for Card Games

Matias Castillo, Benjamin Goeing, Jesper Westell

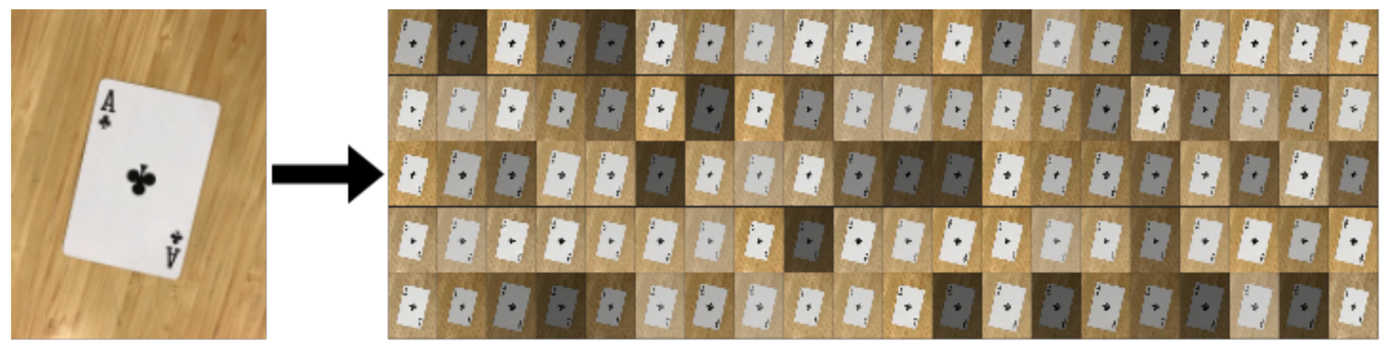
Project Objective

- For our joint CS229/CS221 project, we are developing a program that can play real world card games against a human player
- The goal of our 229 project is to develop a computer vision method, that can recognize playing cards lying on a table with 100% accuracy
- This is a 52-class classification problem



Data Augmentation and Engineering

- For the purpose of this project, we have created our own dataset by taking 10 pictures of each playing card lying on a table
- We then wrote a script to augment every image, using different combinations of zoom, brightness, contrast, rotation etc.



- This resulted in 1,050 examples per class for a total of 54,600 images, which we divided into 90% train, 5% dev, and 5% test
- As the training set was too large to fit into memory, we trained using random mini-batches, that were sequentially read into memory and also decided to downsize the images

References

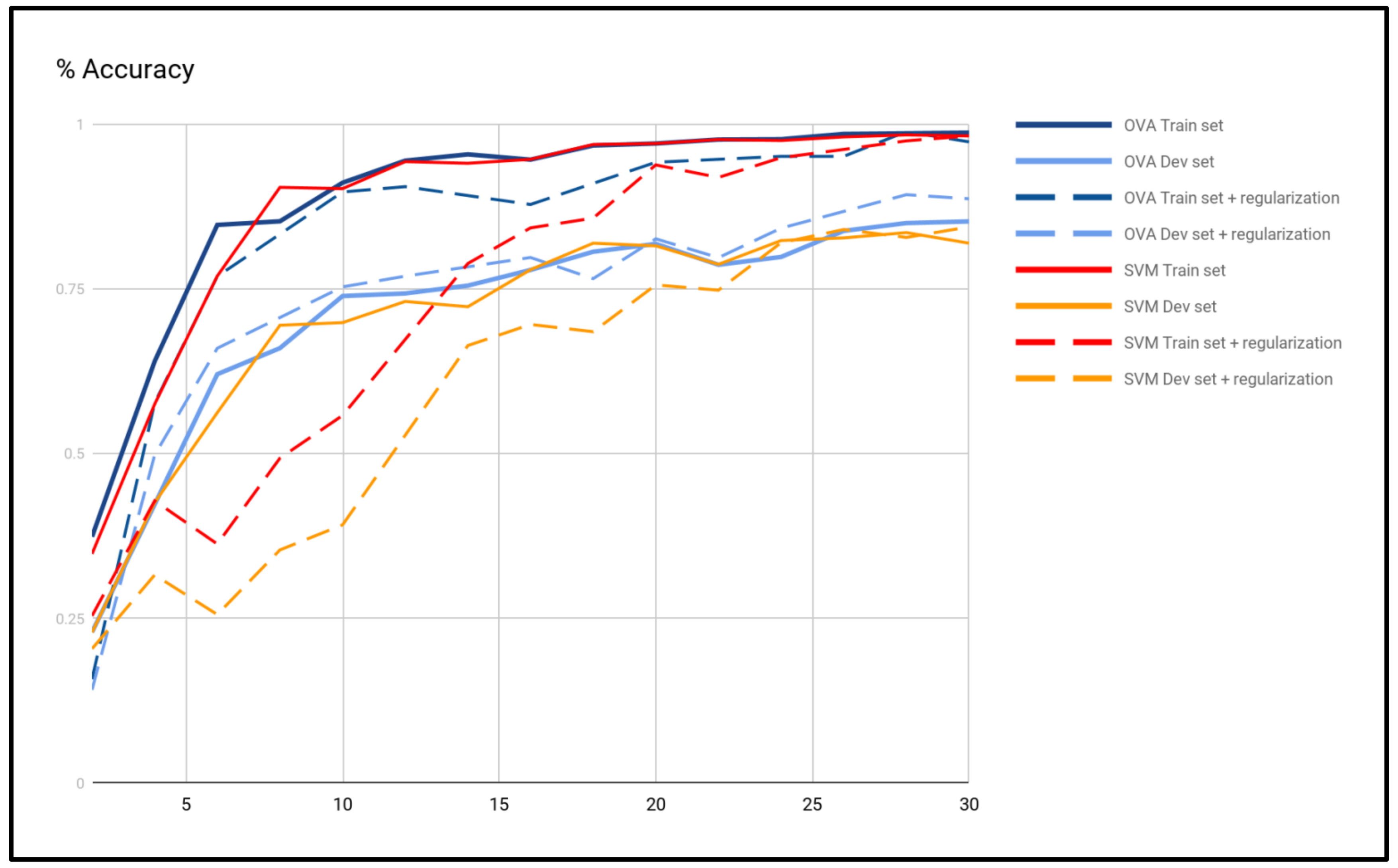
- DeepLearning.ai / CS230 course materials

Classical Machine Learning Methods

We tested two classical machine learning methods, using scikit-learn:

- **Multi-class OVA classification**, using log loss, where we produced a list of classifiers $f_k, k \in \{1...52\}$ and
$$\hat{y} = \arg \max_{k \in \{1...52\}} f_k(x)$$
- **Multi-class OVA classification with L2 Regularization**, using lambda 0.1
- **Multi-class SVM**, using hinge loss
- **Multi-class SVM with L2 Regularization**, using lambda 0.1

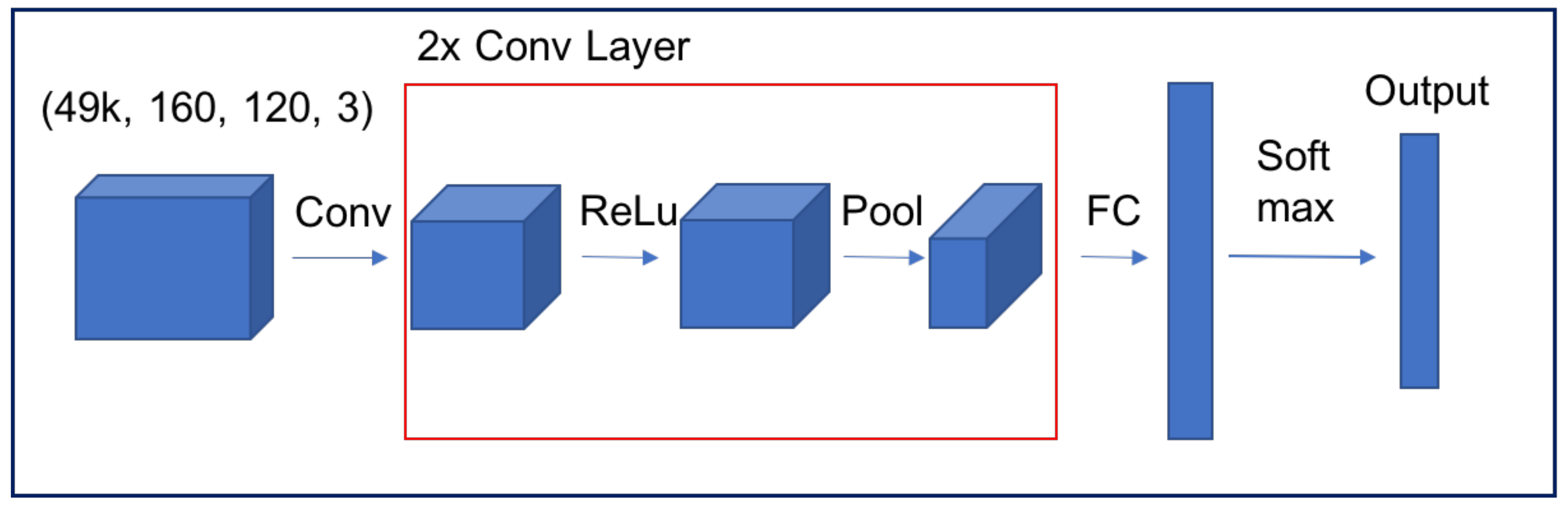
Results from classical methods



- The multi-class OVA performs better than the SVM model (85% vs. 82% dev set accuracy)
- There are still noticeable overfitting issues with both models
- Applying L2 regularization improved dev set accuracy in both models (from 85% to 89% in OVA and from 82% to 85% in SVM).

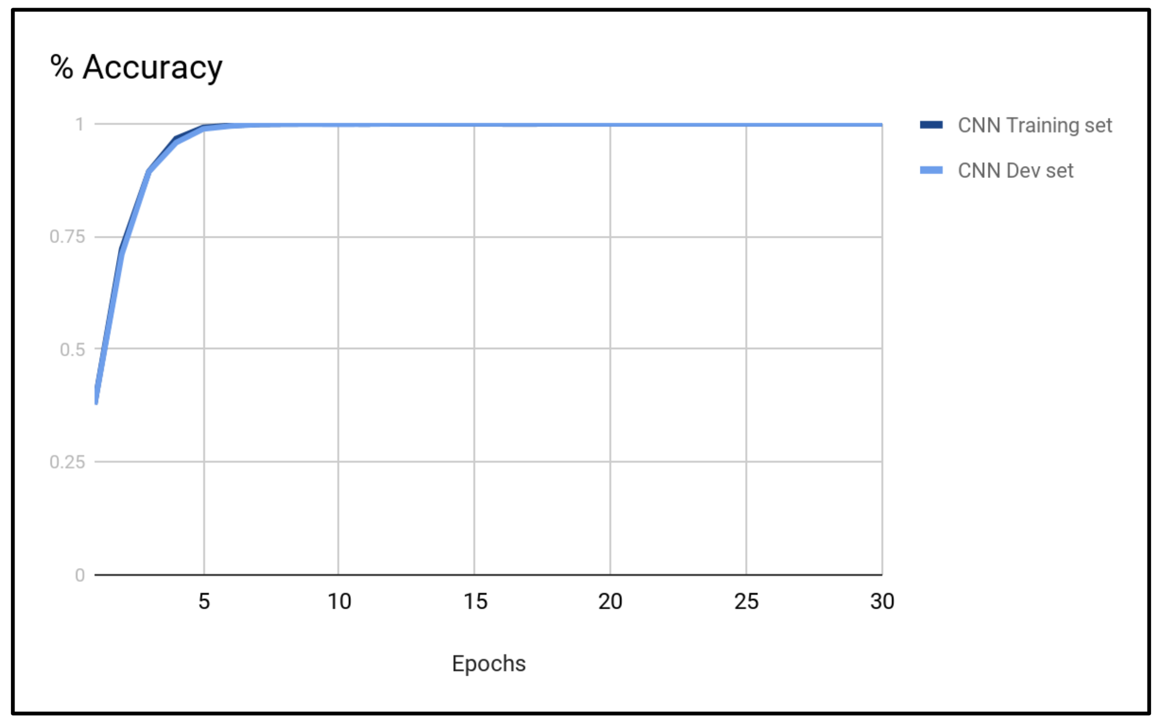
Training a custom CNN from scratch:

- In order to reach our goal of achieving 100% accuracy, we decided to design and train a custom CNN using Tensorflow from scratch:



- We used the cross-entropy loss as a cost function:
$$J(\theta) = -\sum_i y_i ln(\hat{y}_i)$$
- We decided to use a ReLu activation function, max pooling with window sizes 6x6 and 4x4 and strides 6 and 4 respectively, as well as a "same" padding
- Filters used are [4,4,3,8] and [2,2,8,16]

Results Convolutional Neural Net



- We managed to achieve 100.0% train, dev and test accuracy with the CNN after 17 epochs

FUTURE WORK

- All of our card images are coming from the same deck that we purchased
- In future work, we plan to use more varied card sets (e.g. with different themes)
- In addition, we are planning to train a model that can recognize multiple cards at the same time