Skeletal Insights

Yoga pose classification using deep learning

Shahbaz Chaudhary, Emily Coppess, Jay Ong
Master Of Analytics Capstone Project
University Of Chicago

Executive Summary

There is a gap in the virtual learning technology for yoga - feedback

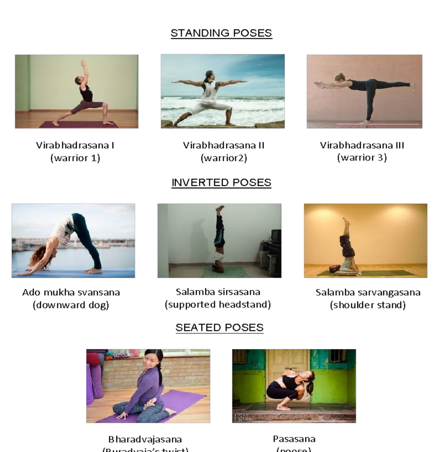
To give feedback, an application needs to be able to identify user actions

We show that an end-to-end deep learning classifier is the best method for a machine identifying yoga poses

Expectations: Implicit versus explicit feature identification?

	Pros	Cons
Pose Extractor: Explicitly identifying features	· Filters out noise · Low data demands · Easy to generate insights	· Compounds errors · Relatively new DL approach
Image Classifier: Implicitly finds features	· Simpler pipeline · Recognizes more complex patterns · Better accuracy	· High data demands · Difficult to generate insights

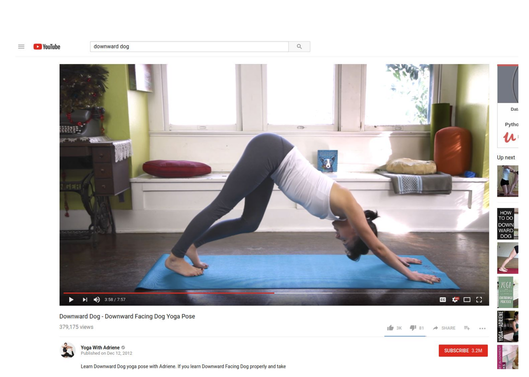
Google Images searches must use Sanskrit



- Searching using English keywords produces noisy results
  - Lots of manual cleaning
- Using Sanskrit terms produces far cleaner results
  - About 100-200 images per pose

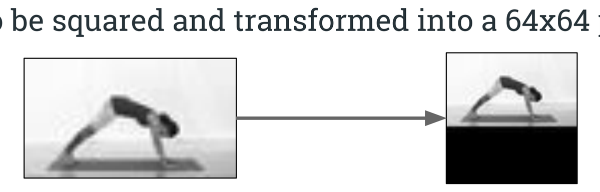
YouTube video frames as data source is not scalable

- We identified and recorded timestamps for poses in yoga instruction videos
- Extracted frames created a lot of redundant images
- Youtube frames produced a quarter of our dataset

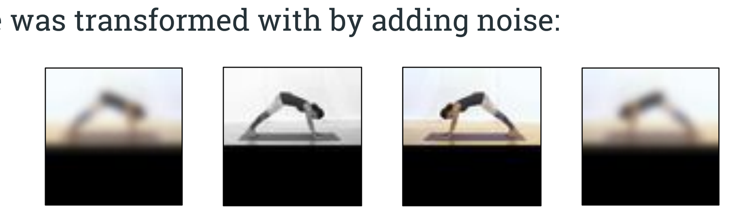


Data underwent two stages of processing

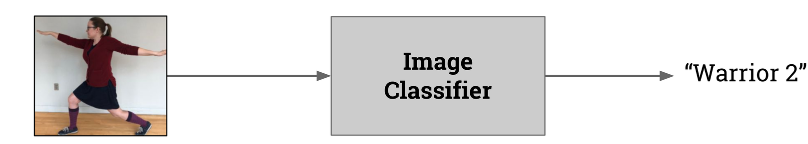
- Images needed to be squared and transformed into a 64x64 pixel array



- Each image was transformed with by adding noise:

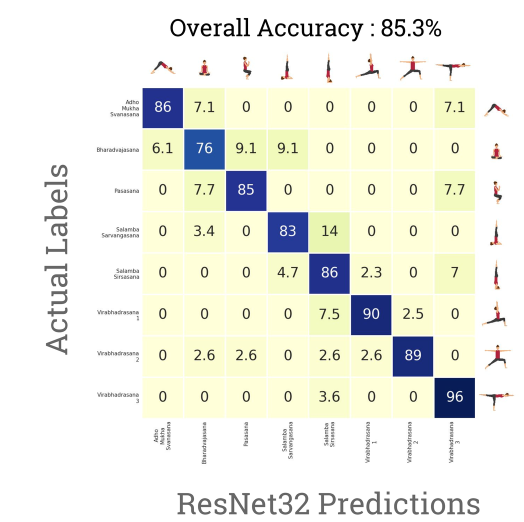


We explored increasingly sophisticated deep learning architectures to improve classification



Models Attempted:
- Neural Network with Fully Connected Layers
- Deep Convolutional Nets (with,w/o pooling) [LeCun, 1998]
- Google's Inception v4 [Szegedy et al., 2016]
- Deep ResNet (56, 101 layers)
- ResNet [He et al. ,2015] (32 layers)

One-step model reliably predicts any given pose



Two-Step Winning Model: Random Forest

- Random Forest: 0.41 accuracy
- Processing time: 42 minutes

Accuracy Scores

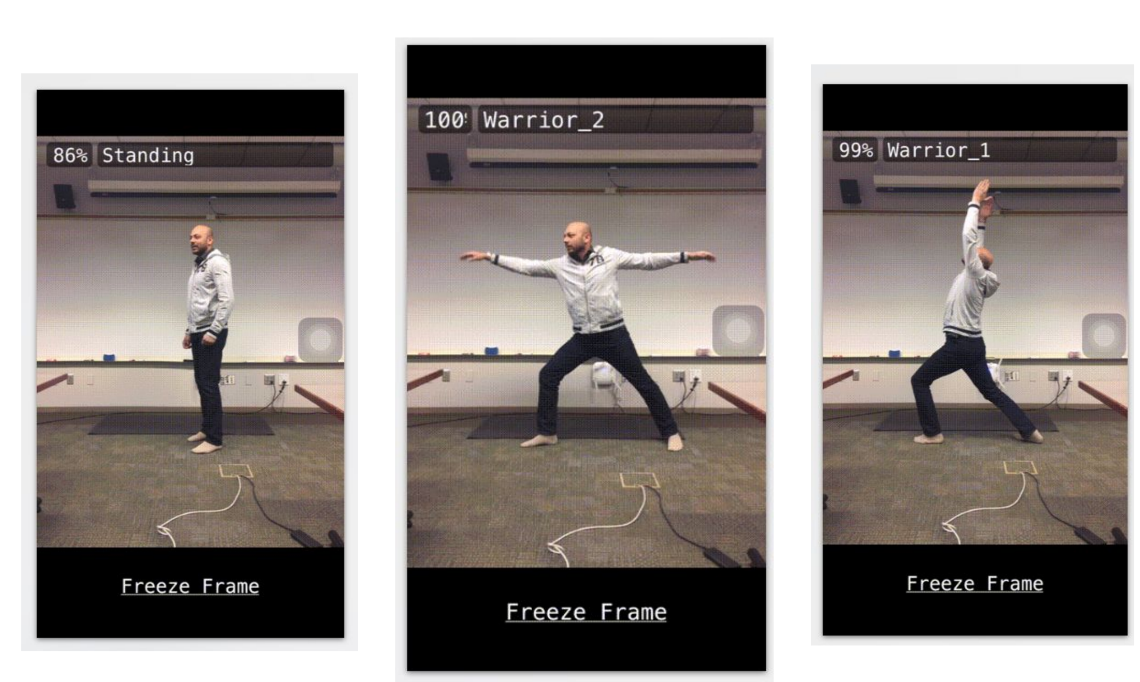
	Joint Coordinates		Joint Angles	
	Random Forest	SVM	Random Forest	SVM
Original: 1,236 images	0.39	0.28	0.26	0.25
Original + transformed: 6,180 training images	0.41	0.29	0.31	0.25

Hyperparameters and infrastructure used

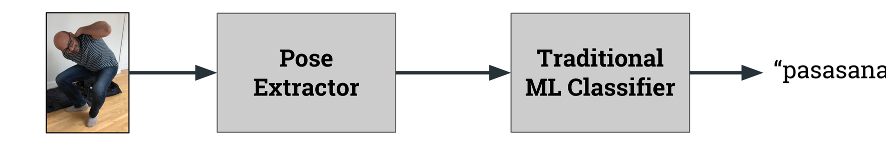
Loss function: **Categorical Cross-Entropy** (a.k.a. Multi-Class Log Loss)

Hyperparameters:
- Backprop Optimization: **Stochastic Gradient Descent with Momentum**
- Learning Rate : **0.01-0.1**
- Number of Iterations: **50-200** epochs (**1 forward & backward pass** for ALL of training set)
- Regularization: **Dropout** (a.k.a. ensemble), **Batch Normalization**

Infrastructure:
- Research Computing Center GPU Clusters*
- Google Colab GPU Clusters*

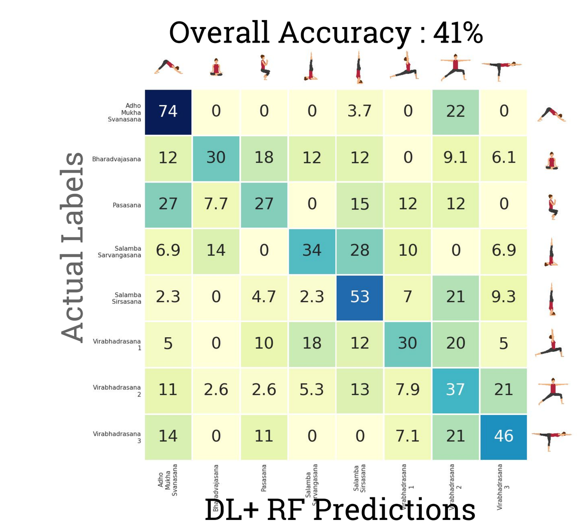


Two-Step Method combines DL with traditional ML



- Use pre-trained model "DeeperCut" to convert images to joint location data
- Train classical machine learning to classify poses

Two step method gets easily confused

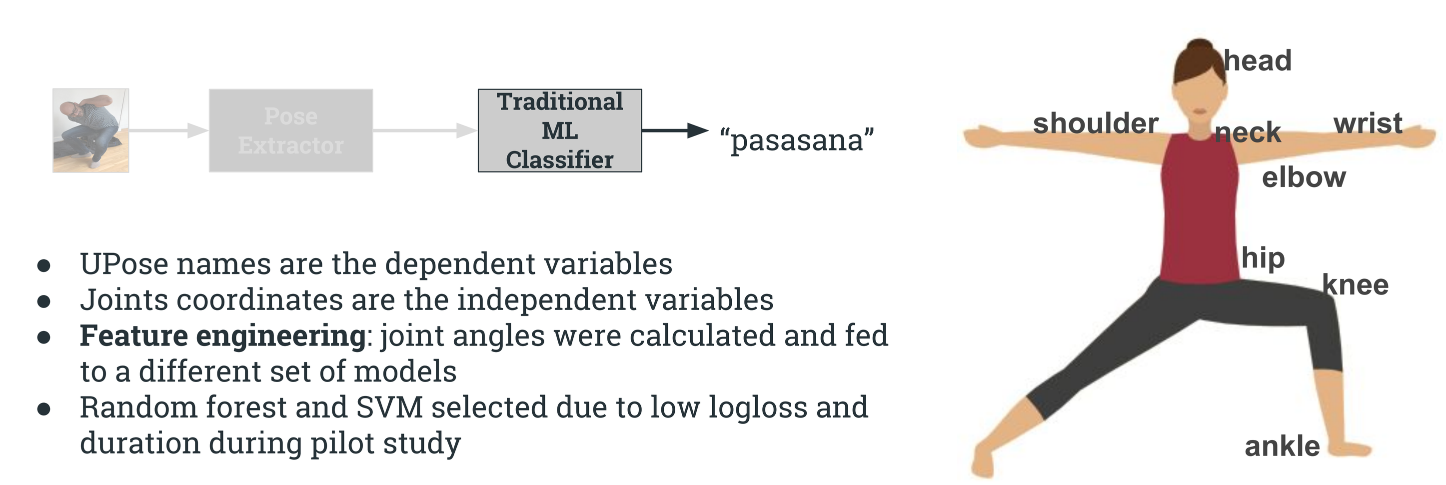


The best model for one-step classification was a ResNet with 32 layers

	ConvNet	ConvNet No Pooling	ConvNet 11 Layers	ResNet32	ResNet56
Accuracy	0.625	0.679	0.645	0.853	0.826
Training time seconds	1,297.77	666.80	4,566.34	1,962.29	3,564.10

- ResNet allowed us to train more layers faster.
- ResNet32 was the most predictive model
- Prediction rate was 0.0174 seconds per image (57 Frames Per Second)

Joint coordinates are fed into a classifier



- UPose names are the dependent variables
- Joints coordinates are the independent variables
- **Feature engineering**: joint angles were calculated and fed to a different set of models
- Random forest and SVM selected due to low logloss and duration during pilot study

2D projection (T-SNE) shows images are not clustered