SPOT

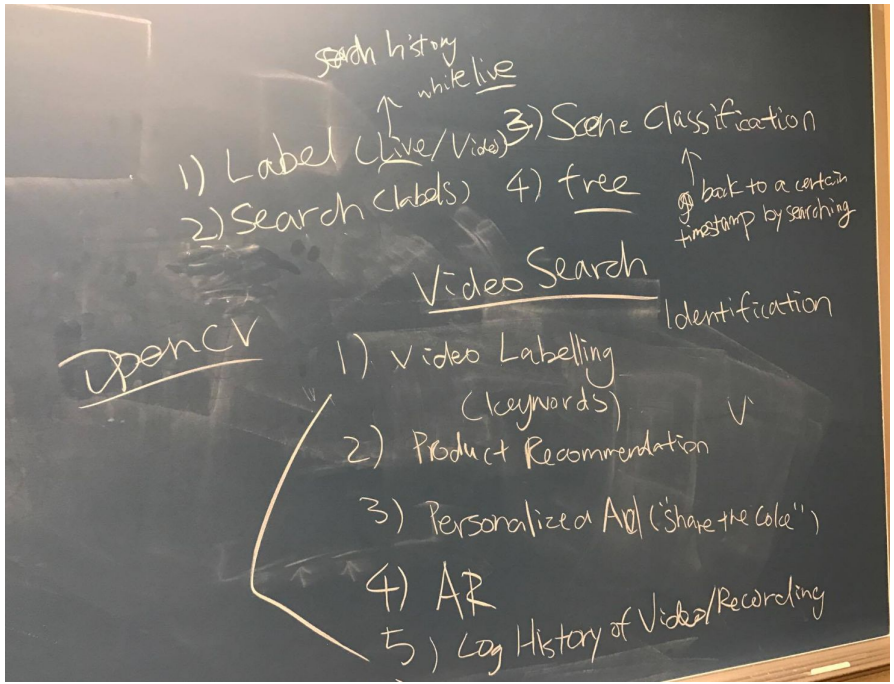
Data-X

Kate Xu, Nikita Vemuri, Prathyusha Chragondla, Michael Tu, Mark Annevelink

Problem We Are Solving

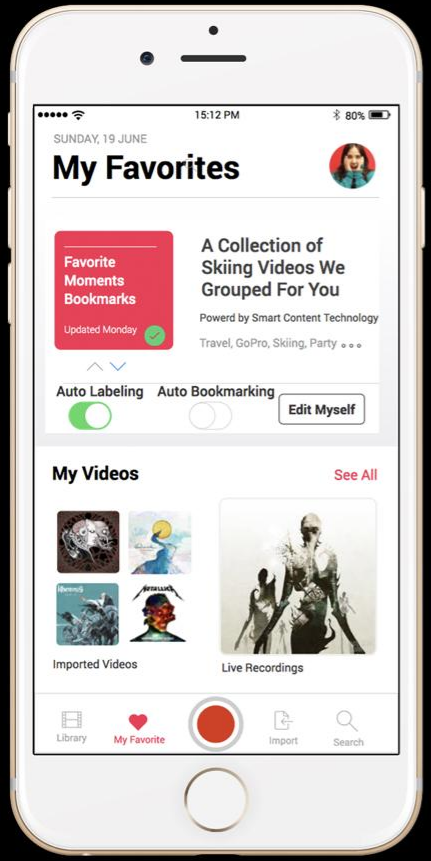
Real Time/Video Image Classification

- Video captures a cross-section of our society.
- And major advances in analyzing and understanding video have the potential to touch all aspects of life from learning and communication to entertainment and play.
- YouTube labeling and classification
- Google the content of a video
- Real time analysis for drones
- etc...



Our Brainstorming Session

Intended User Interface



IEOR 190D/290 LOW TECH PROJECT DEMO

Video Content Search Engine

Team Member: Gan Tu, Kate Xu, Mark Annevelink, Nikita Vemuri, Prathyusha Charagondla

About the Project:
Videos capture significant memories in our lives and have become a indispensable aspect of our society. So, we decided to build this app to advance the state of art in video understanding by providing an analysis tool for videos, potentially positively impacting all aspects of lives from learning and communication to entertainment and play.

CONTENT LABELING (LIVE/VIDEO)

Our product recommends relevant labels and tags for both live streams and videos based on their contents

SMART SEARCH

We provide you a search engine that helps you find videos you need by searching through both video titles and the footages themselves

SUGGESTIVE BOOKMARKS

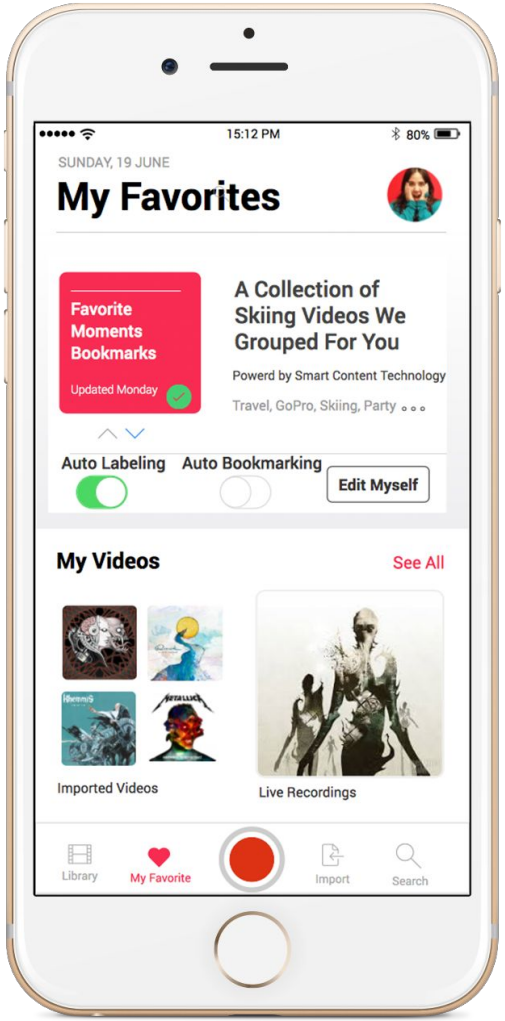
We automatically label breakpoints for your videos so you know where your favorite moments are in the video, instead of you going through the entire footage to find them.

MOBILE

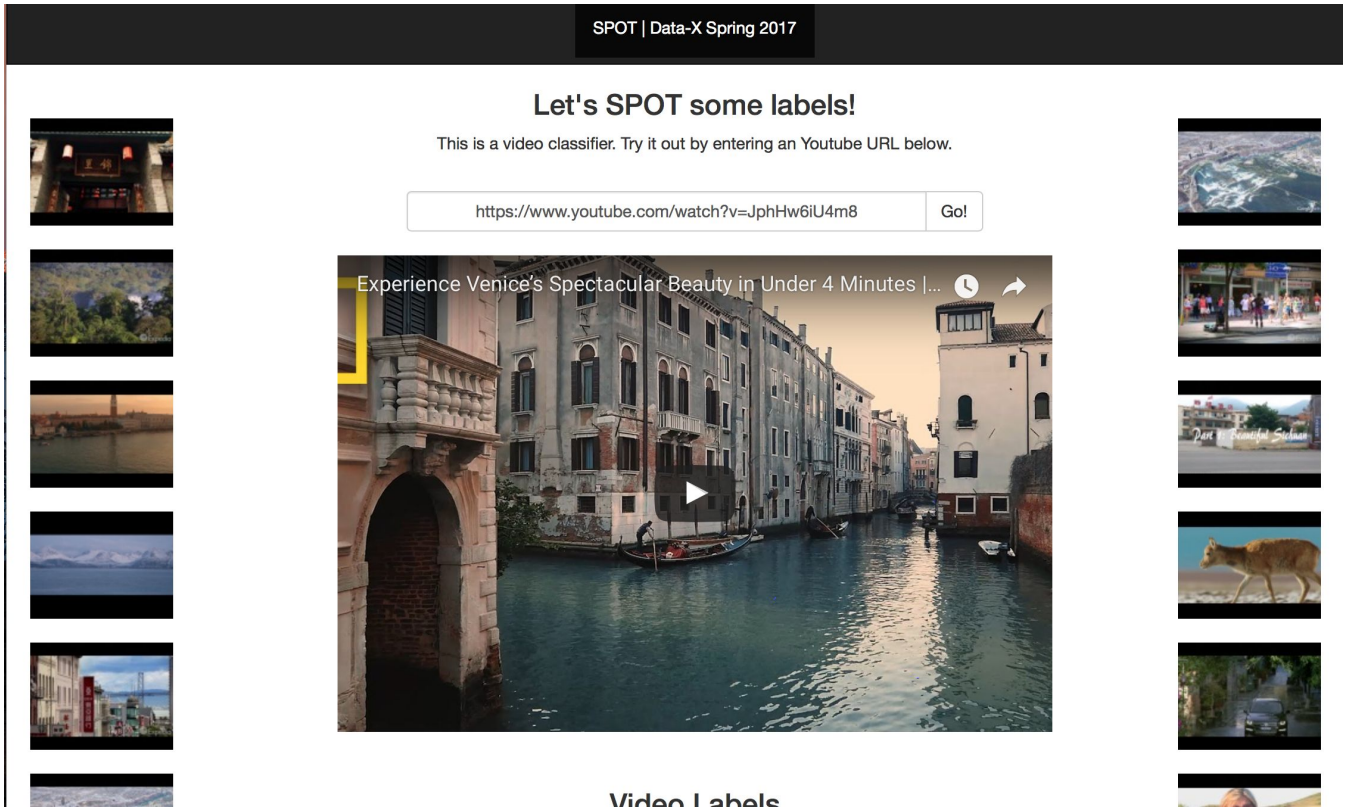
No complicated software to install and no need for a laptop. Our app is mobile friendly so you can label your videos wherever you are whenever you want.

Pivot: Conversion from mobile to web-app

- OpenCV is better suited for our task of breaking videos and live recordings into frames on laptop
- Limited Documentation for OpenCV iOS.
- No mobile development experience on the team


15:12 PM
80%
SUNDAY, 19 JUNE
My Favorites
Favorite Moments Bookmarks
Updated Monday
A Collection of Skiing Videos We Grouped For You
Powerd by Smart Content Technology
Travel, GoPro, Skiing, Party ...
Auto Labeling
Auto Bookmarking
Edit Myself
My Videos
See All
Imported Videos
Live Recordings
Library
My Favorite
Import
Search

Before….


SPOT | Data-X Spring 2017
Let's SPOT some labels!
This is a video classifier. Try it out by entering an Youtube URL below.
https://www.youtube.com/watch?v=JphHw6iU4m8
Go!
Experience Venice's Spectacular Beauty in Under 4 Minutes |...
Video Labels

After....

Our Journey To The Solution

--- Journey of Pivots

Before we had the Image Recognition Lecture…

Easier Problem -- Digit Recognition

The Street View House Numbers (SVHN) Dataset

Reading Digits in Natural Images with Unsupervised Feature Learning

Yuval Netzer[1], Tao Wang[2], Adam Coates[2], Alessandro Bissacco[1], Bo Wu[1], Andrew Y. Ng[1,2]
{yuvaln,bissacco,bowu}@google.com
{twangcat,acoates,ang}@cs.stanford.edu

[1]Google Inc., Mountain View, CA
[2]Stanford University, Stanford, CA

Abstract

Detecting and reading text from natural images is a hard computer vision task that is central to a variety of emerging applications. Related problems like document character recognition have been widely studied by computer vision and machine learning researchers and are virtually solved for practical applications like reading

Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks

Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet
Google Inc., Mountain View, CA
[goodfellow,yaroslavvb,julianibarz,sacha,vinayshet]@google.com

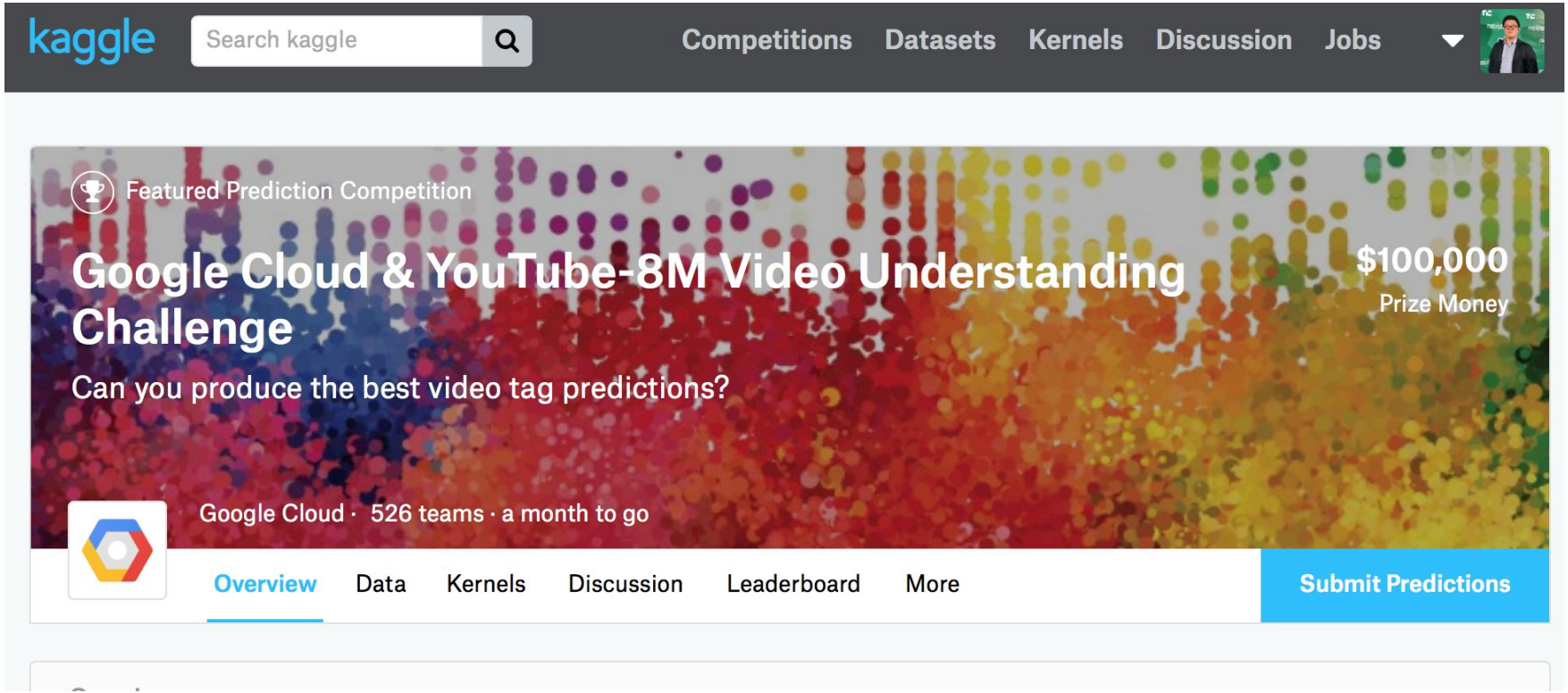
Abstract

Recognizing arbitrary multi-character text in unconstrained natural photographs is a hard problem. In this paper, we address an equally hard sub-problem in this domain viz. recognizing arbitrary multi-digit numbers from Street View imagery. Traditional approaches to solve this problem typically separate out the localization, segmentation, and recognition steps. In this paper we propose a unified approach that integrates these three steps via the use of a deep convolutional neural network that operates directly on the image pixels. We employ the DistBelief (Dean *et al.*, 2012) implementation of deep neural networks in order to train large, distributed neural networks on high quality images. We find that the performance of this approach increases with the depth of the convolutional network,

Idea: Instead of real time recognizing many things, maybe just real time recognize digits?

Then.. we had image recognition lecture !

And also Kaggle released something new...


kaggle
Search kaggle
Competitions
Datasets
Kernels
Discussion
Jobs
Featured Prediction Competition
Google Cloud & YouTube-8M Video Understanding Challenge
Can you produce the best video tag predictions?
$100,000
Prize Money
Google Cloud · 526 teams · a month to go
Overview
Data
Kernels
Discussion
Leaderboard
More
Submit Predictions

Awesome! We have the data now?

File descriptions

- **video-level data**
  - Available on Google Cloud at gs://us.data.yt8m.org/1/video_level/train , gs://us.data.yt8m.org/1/video_level/validate , and gs://us.data.yt8m.org/1/video_level/test
  - You can access the data on Google Cloud, or download to your local computer with instructions here
  - Total size of 31GB
  - Each video has

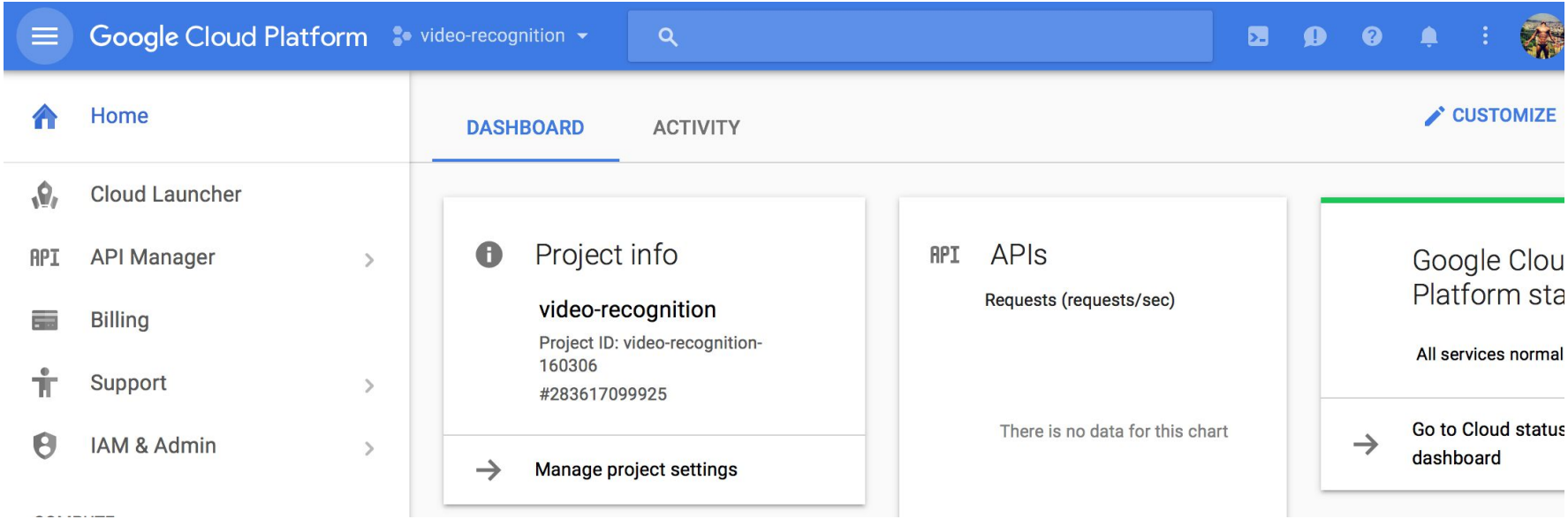
a. "video_id": unique id for the video, in train set it is a Youtube video id, and in test/validation they are anonymized
b. "labels": list of labels of that video
c. "mean_rgb": float array of length 1024
d. "mean_audio": float array of length 128

  - Files are in TFRecords format, TensorFlow python readers are available in the github repo, and simple reader available in Kaggle Kernels

- **frame-level data**
  - Available on Google Cloud at gs://us.data.yt8m.org/1/frame_level/train , gs://us.data.yt8m.org/1/frame_level/validate , and gs://us.data.yt8m.org/1/frame_level/test
  - You can access the data on Google Cloud, or download to your local computer with instructions here
  - Total size of 1.71TB (Large file warning!)
  - Each video has

File descriptions

- **video-level data**
  - Available on Google Cloud at gs://us.data.yt8m.org/1/video_level/train , gs://us.data.yt8m.org/1/video_level/validate , and gs://us.data.yt8m.org/1/video_level/test
  - You can access the data on Google Cloud, or download to your local computer with instructions here
  - Total size of 31GB



README.md	Remove instructic
__init__.py	Setup.py autogen
average_precision_calculator.py	Add support for d
cloudml-4gpu.yaml	Add support for m
cloudml-gpu-distributed.yaml	Fixing a bug that
cloudml-gpu.yaml	Add support for m
convert_prediction_from_json_to_...	Add support for b
eval.py	Add support for m
eval_util.py	Add support for d
export_model.py	Add support for m
frame_level_models.py	Merge pull reques
inference.py	Add support for m
losses.py	Add support for d
mean_average_precision_calculat...	Add support for d
model_utils.py	Add Moe and DBo
models.py	Add Moe and DBo
readers.py	Add support for m
train.py	Removed useless
utils.py	Add support for m

22 lines (18 sloc) | 824 Bytes

```
1  # Copyright 2016 Google Inc. All Rights Reserved.
2  #
3  # Licensed under the Apache License, Version 2.0 (the "License");
4  # you may not use this file except in compliance with the License.
5  # You may obtain a copy of the License at
6  #
7  #      http://www.apache.org/licenses/LICENSE-2.0
8  #
9  # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS-IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14
15 """Contains the base class for models."""
16
17 class BaseModel(object):
18   """Inherit from this class when implementing new models."""
19
20   def create_model(self, unused_model_input, **unused_params):
21     raise NotImplementedError()
```

We looked into existing researches

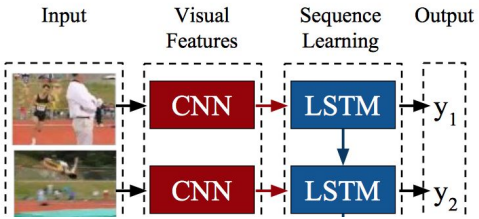
Long-term Recurrent Convolutional Networks for Visual Recognition and Description

Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell

Abstract—
Models based on deep convolutional networks have dominated recent image interpretation tasks; we investigate whether models which are also recurrent are effective for tasks involving sequences, visual and otherwise. We describe a class of recurrent convolutional architectures which is end-to-end trainable and suitable for large-scale visual understanding tasks, and demonstrate the value of these models for activity recognition, image captioning, and video description. In contrast to previous models which assume a fixed visual representation or perform simple temporal averaging for sequential processing, recurrent convolutional models are "doubly deep" in that they learn compositional representations in space and time. Learning long-term dependencies is possible when nonlinearities are incorporated into the network state updates. Differentiable recurrent models are appealing in that they can directly map variable-length inputs (e.g., videos) to variable-length outputs (e.g., natural language text) and can model complex temporal dynamics; yet they can be optimized with backpropagation. Our recurrent sequence models are directly connected to modern visual convolutional network models and can be jointly trained to learn temporal dynamics and convolutional perceptual representations. Our results show that such models have distinct advantages over state-of-the-art models for recognition or generation which are separately defined or optimized.

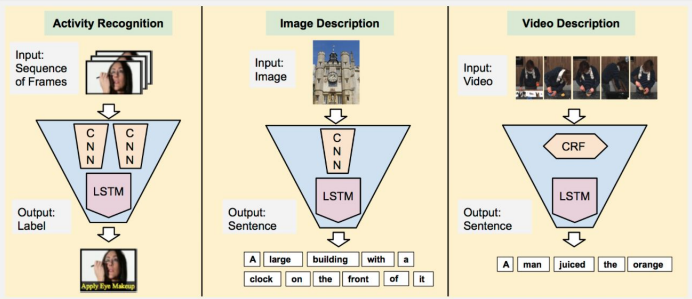
1 Introduction

Recognition and description of images and videos is a fundamental challenge of computer vision. Dramatic progress has been achieved by supervised convolutional neural network (CNN) models on image recognition tasks, and a number of extensions to process video have been recently proposed. Ideally, a video model should allow processing of variable length input sequences, and also provide for variable length outputs, including generation of full-length sentence descriptions that go beyond conventional one-versus-all prediction tasks. In this paper we propose



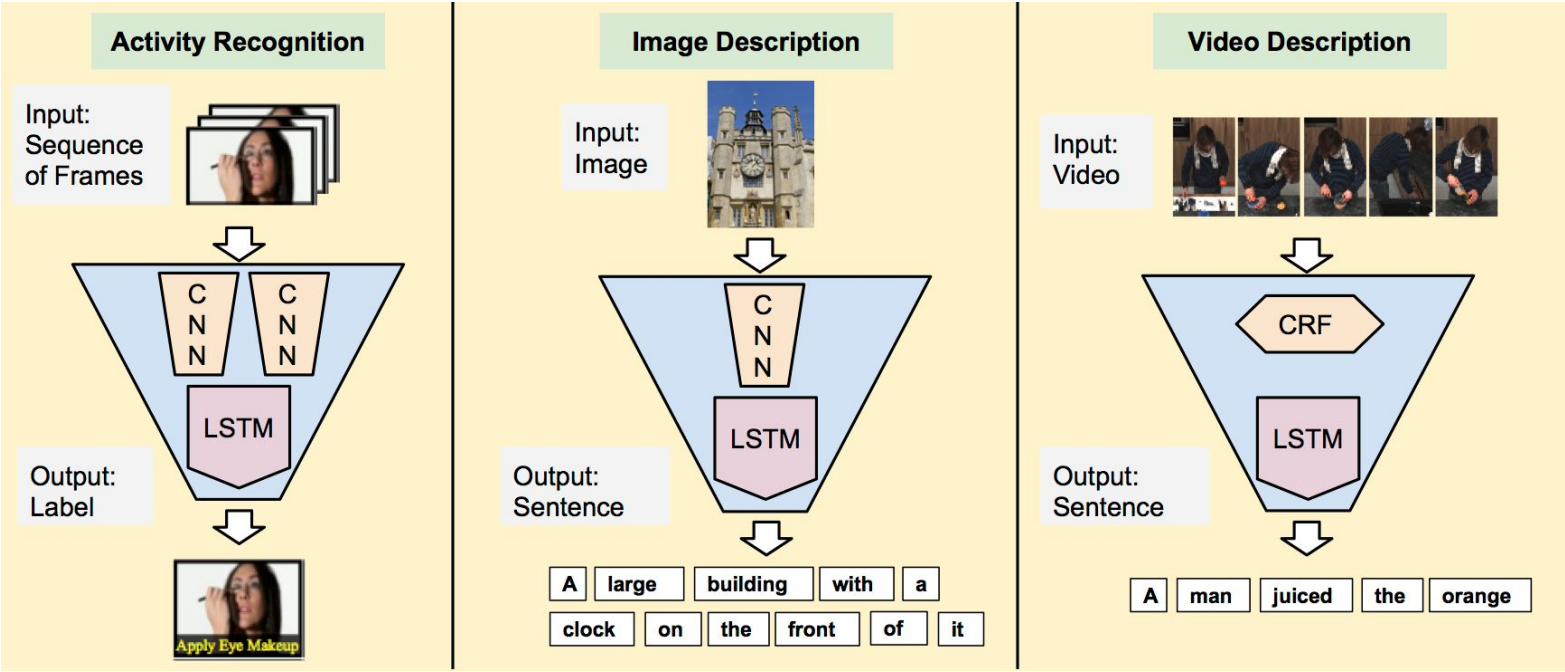
Long-term Recurrent Convolutional Networks

This is the project page for Long-term Recurrent Convolutional Networks (LRCN), a class of models that unifies the state of the art in visual and sequence learning. LRCN was accepted as an **oral presentation** at CVPR 2015. See our arXiv report for details on our approach.



LRCN - Combining LSTMs and CNNS

Model


Activity Recognition
Input: Sequence of Frames
CNN
CNN
LSTM
Output: Label
Apply Eye Makeup
Image Description
Input: Image
CNN
LSTM
Output: Sentence
A large building with a clock on the front of it
Video Description
Input: Video
CRF
LSTM
Output: Sentence
A man juiced the orange

Take long time to train??
What should we do?

Inspired by HW7: Pre-Trained Weights

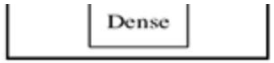
Part 2: Extract bottleneck features from the data set

In the second part you will use the pre-trained VGG network structure (loading in the pretrained VGG16 ImageNET weights). Then you will run your data set through that CNN ones to extract the image features.

A good explanation on how this works (rewritten from source: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

Using the bottleneck features of a pre-trained network: 90% accuracy in 1 min (GPU) / 10 mins (CPU)

In Part 2 we leverage a network pre-trained on a large dataset. Such a network would have already learned features that are useful for most computer vision problems, and leveraging such features would allow us to reach a better accuracy than any method that would only rely on the available data.



Strategy to extract bottleneck features: We will only instantiate the convolutional part of the model, everything up to the fully-connected layers. We will then run this model on our training, validation and test data once, recording the output (the "bottleneck features" from the VGG16 model: the last activation maps before the fully-connected layers) in two numpy arrays. The reason why we are storing the features offline rather than adding our fully-connected model directly on top of a frozen convolutional base and running the whole thing, is computational effiency. Running VGG16 is expensive, especially if you're working on CPU, and we want to only do it once. Note, therefore we will not use data augmentation.

Found Pre-trained Weights on Video Frames

This page contains all the weights needed to train the video models described in the paper Long-term Recurrent Convolutional Networks for Visual Recognition and Description.

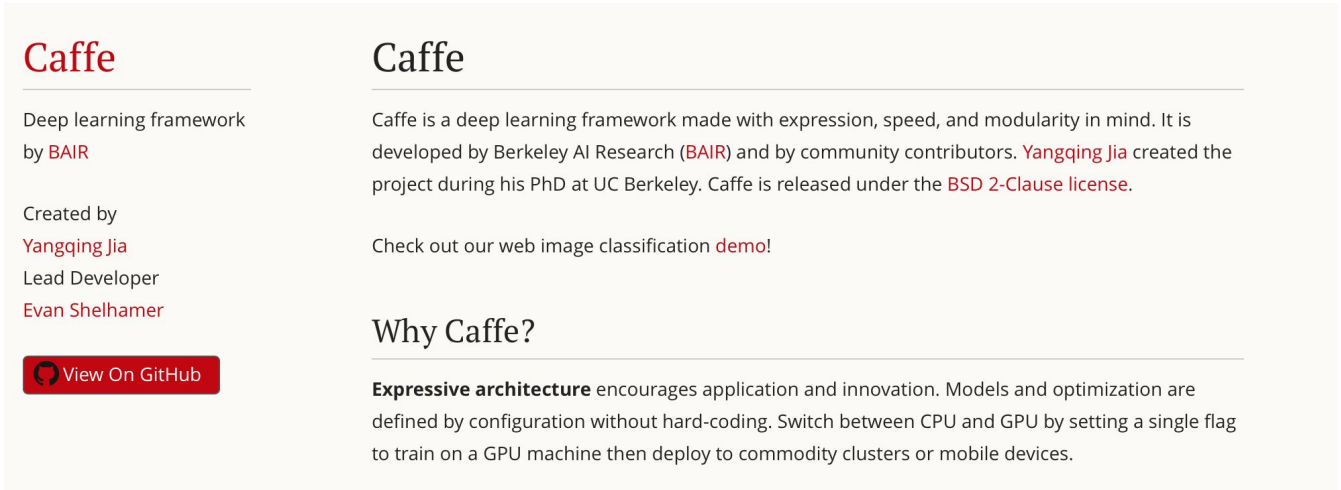
The following model was used to pretrain the single frame RGB and flow video models. It is a hybrid between the reference *Caffe* net [1] and the network used by Zeil Fergus [2]:
[caffe_imagenet_hyb2_wr_rc_solver_sqrt_iter_310000]

The following are single frame models trained on the first split of UCF101:
Single frame RGB model:
[Single frame RGB model]
Single frame flow model:
[Single frame flow model]

The following are the final LRCN video models trained on the first split of UCF101:
LRCN RGB model:
[LSTM RGB model]
LRCN flow model:
[LSTM flow model]

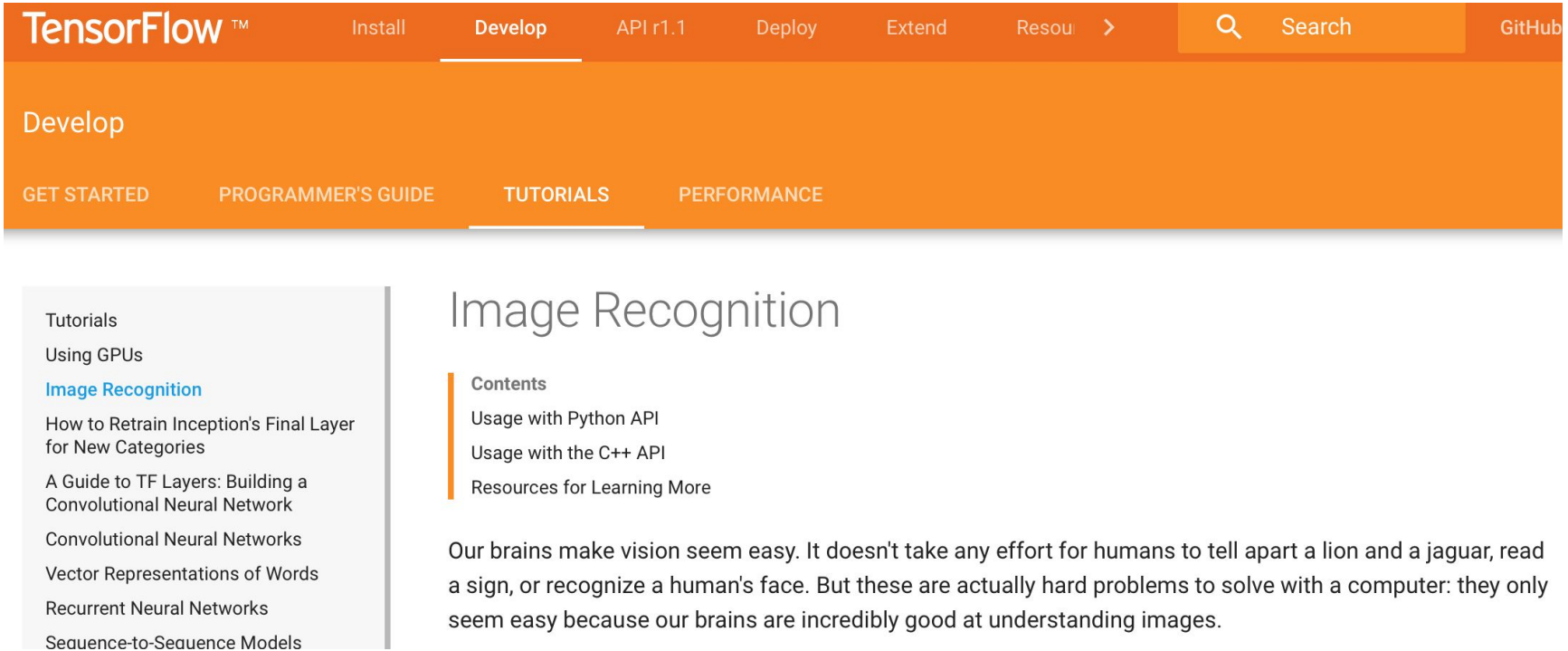
[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM MM, 2014.
[2] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV. 2014.

It uses Caffe…. Mmmm..
OK. Let's learn it….



Nope. Have a hard time installing it…… ;(
(Sad reacts only)

Another Found!
Pre-Trained Weights on ImageNet



Pivot:

Instead of training on videos…

Let’s relax the problem and split videos into frames and use image classification on them

Tensorflow Pre-trained Inception Classifier

models but the results are still hard to reproduce. We're now taking the next step by releasing code for running image recognition on our latest model, Inception-v3.

Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher". For example, here are the results from AlexNet classifying some images:

To compare models, we examine how often the model fails to predict the correct answer as one of their top 5 guesses -- termed "top-5 error rate". AlexNet achieved by setting a top-5 error rate of 15.3% on the 2012 validation data set; Inception (GoogLeNet) achieved 6.67%; BN-Inception-v2 achieved 4.9%; Inception-v3

Tensorflow Pre-trained Inception Classifier

```
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================

"""Simple image classification with Inception.
```

Initial Solution Architecture

Initial Architecture

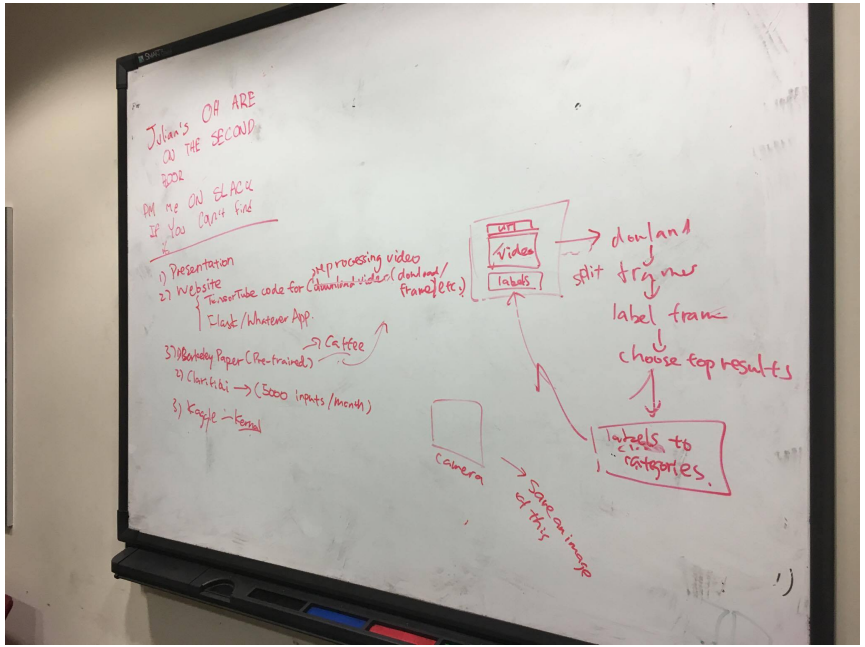
Video Library

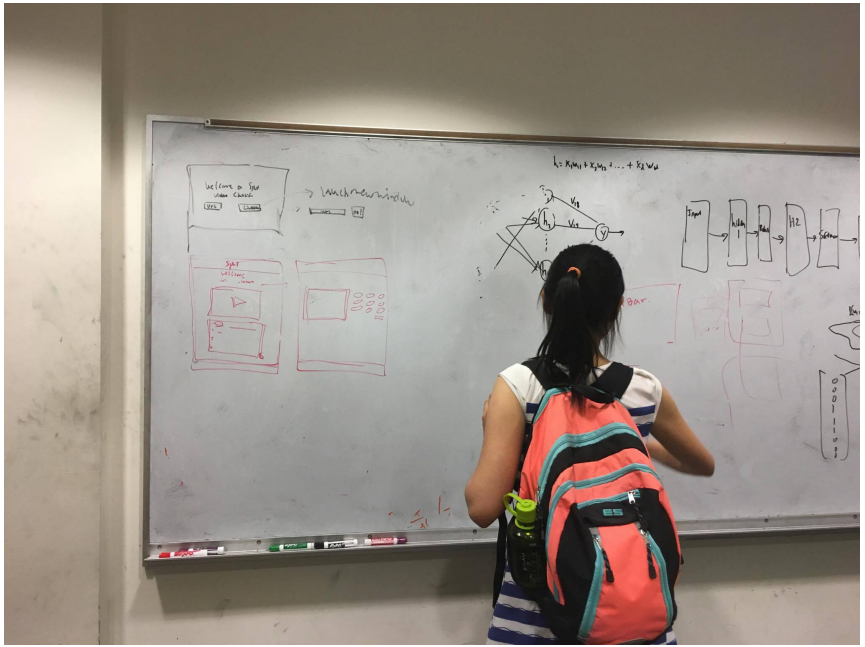
Split Into Frames

Label Frames

Select Top Results

We started working….





Suddenly. An idea popped up!

We shouldn't just sort by the image classification labels alone….

Sometimes, these labels tend to be *<u>too specific</u>*.

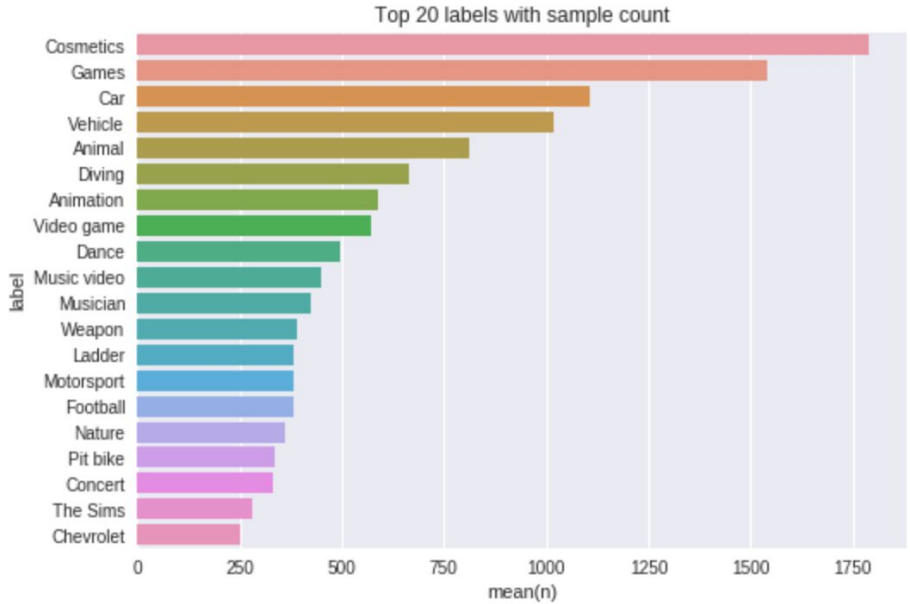
Take a look at Kaggle's Video Classification Labels

In [6]:
```
N = 20

textual_labels_with_counts_all = grouped_data_for(textual_labels)

sns.barplot(y='label', x='n', data=textual_labels_with_counts_all.iloc[0:N, :])
plt.title('Top {} labels with sample count'.format(N))
```

Out[6]: <matplotlib.text.Text at 0x7f18bb8bc3c8>

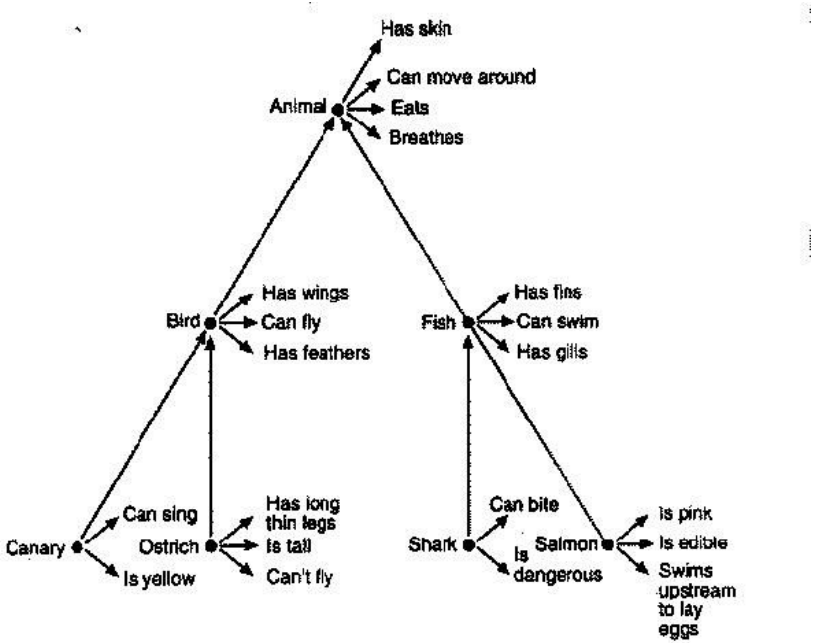


They are generic

A Little Change

For example, instead of outputting “deodorants, lotions, powders, brushes,” we should output “cosmetics”

Approach 1: Transitioning from labels into categories using a Hierarchical Network Model (HNM)



- Understanding text inputs as a function of nodes and properties
- Nodes = major concept such as “Animal”, “Fish”, and “Shark”
- Properties = is an attribute or feature related to the specific node
- Arranged as a hierarchy where the most encompassing nodes are stored on the higher levels

http://www2.fairmontstate.edu/users/ffidura/cogpsy/cpsemorg.html

Better Approach: From HVM to Word2Vec

```
# load model first - it takes some time
# download the Word2Vec Model from https://drive.google.com/file/d/
0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit and put it in the same folder as this file

def loadModel():
    from gensim.models.keyedvectors import KeyedVectors
    model = gensim.models.KeyedVectors.load_word2vec_format('
        GoogleNews-vectors-negative300.bin', binary=True)
    return model

def getCategory (labels, word_input, model = None):
    from gensim.models.keyedvectors import KeyedVectors
    if model == None:
        model = gensim.models.KeyedVectors.load_word2vec_format('
            GoogleNews-vectors-negative300.bin', binary=True)
    for label in labels:
        for word in model.most_similar_cosmul(positive=[label], topn=100):
            if word[0] == word_input:
                return label
    return word_input
```

Output a certain generic categorical label, if our specific image classification labels are synonyms of the categorical label

Final Architecture: Video Labeling

Video Library

Split Into Frames

Label Frames

Select Top Results

Word2Vec

Final Architecture: Real Time Recognition

Live Video

Capture Frames

Save as Images

Label Images

Select Top Results

Word2Vec

Resources Used for Final Website Development

Flask
web development,
one drop at a time

gensim
topic modelling for humans

TensorTube

clarifai

GitHub

OpenCV

kaggle

TensorFlow™

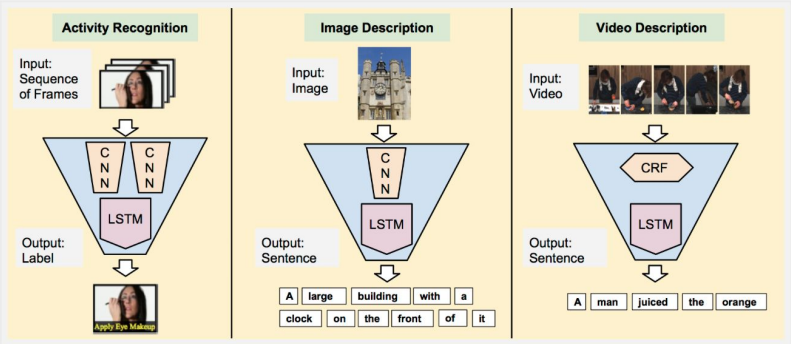
Demo

GitHub Repo: https://github.com/Michael-Tu/data-x-project/tree/master/app

Though we didn’t use the method in this paper due to lack of experience of Caffe, we learned something as well

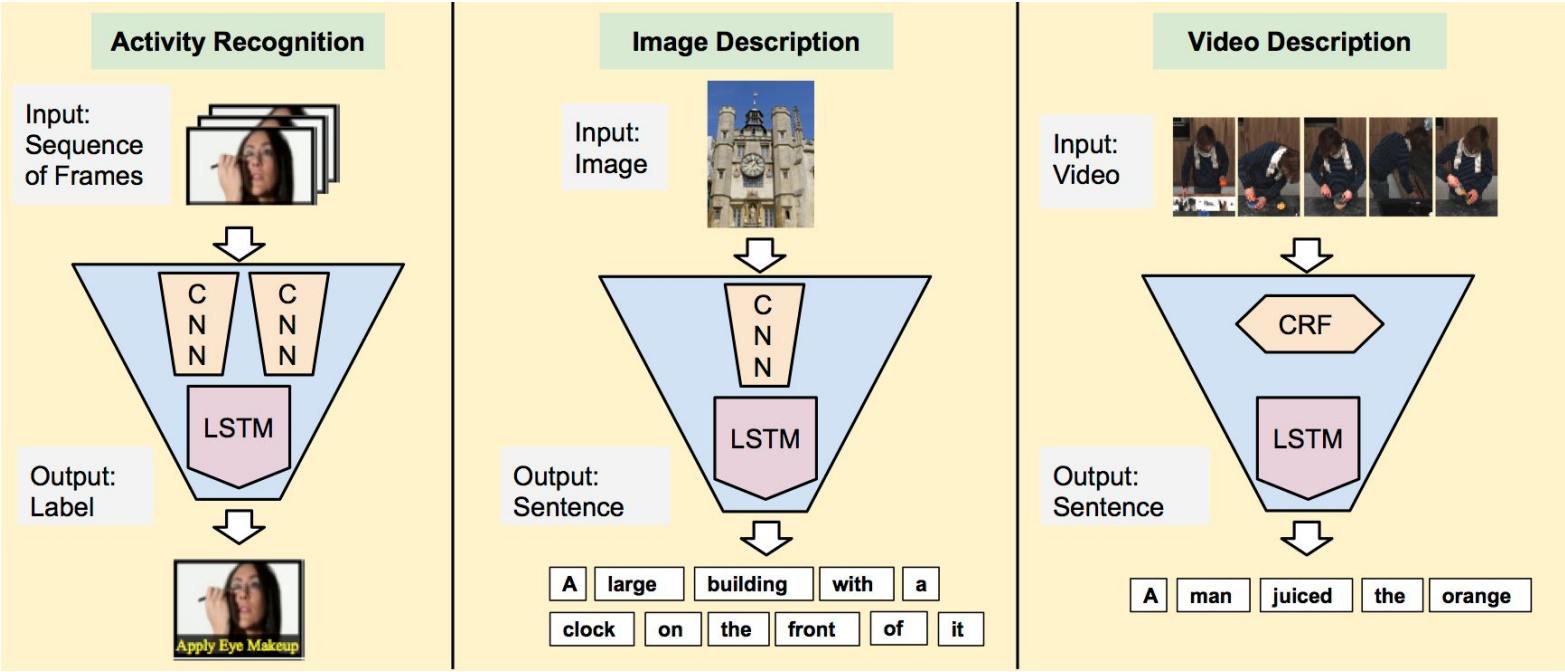
Long-term Recurrent Convolutional Networks

This is the project page for Long-term Recurrent Convolutional Networks (LRCN), a class of models that unifies the state of the art in visual and sequence learning. LRCN was accepted as an **oral presentation** at CVPR 2015. See our arXiv report for details on our approach.



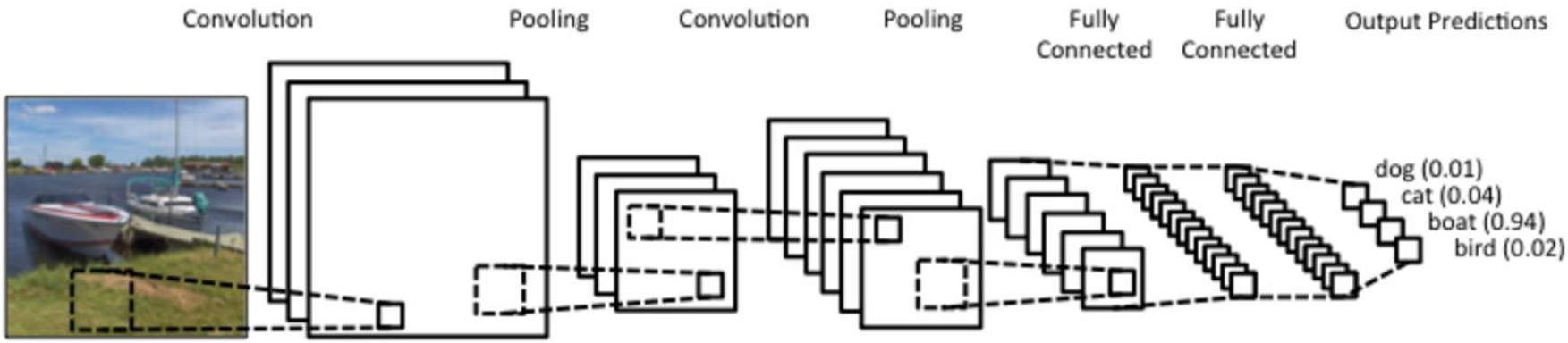
Recap: LRCN - Combining LSTMs and CNNS

Recap: Model


Activity Recognition
Input: Sequence of Frames
CNN
CNN
LSTM
Output: Label
Apply Eye Makeup
Image Description
Input: Image
CNN
LSTM
Output: Sentence
A large building with a clock on the front of it
Video Description
Input: Video
CRF
LSTM
Output: Sentence
A man juiced the orange

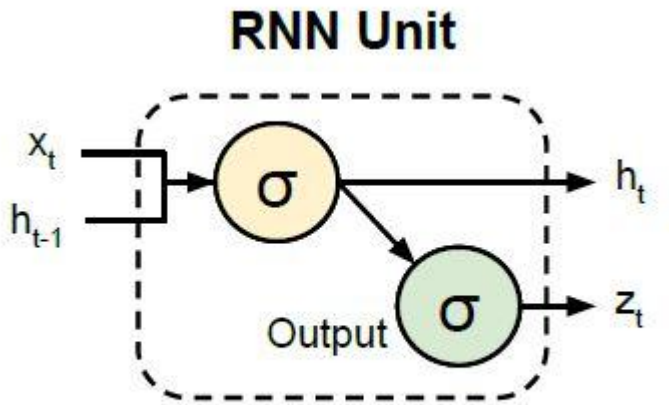
Benefit of CNNs

- Have the ability to capture spatial information in images



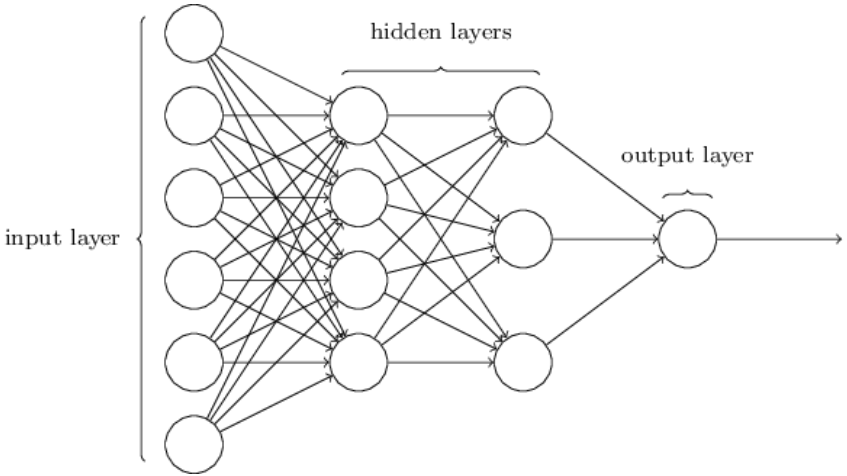
Benefits of RNN

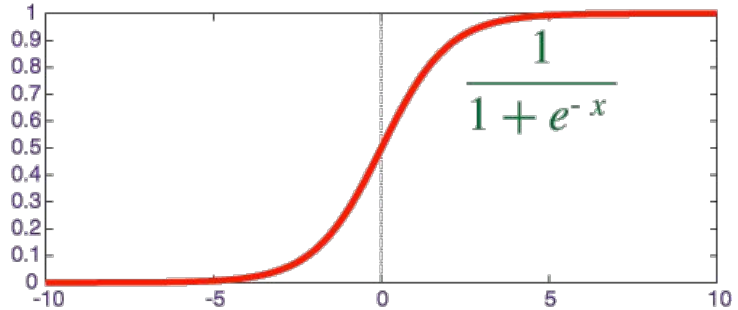
- Deep in the temporal dimension
- Can produce sequential outputs which deviate from basic classification tasks



Drawback of RNNs - Vanishing Gradient

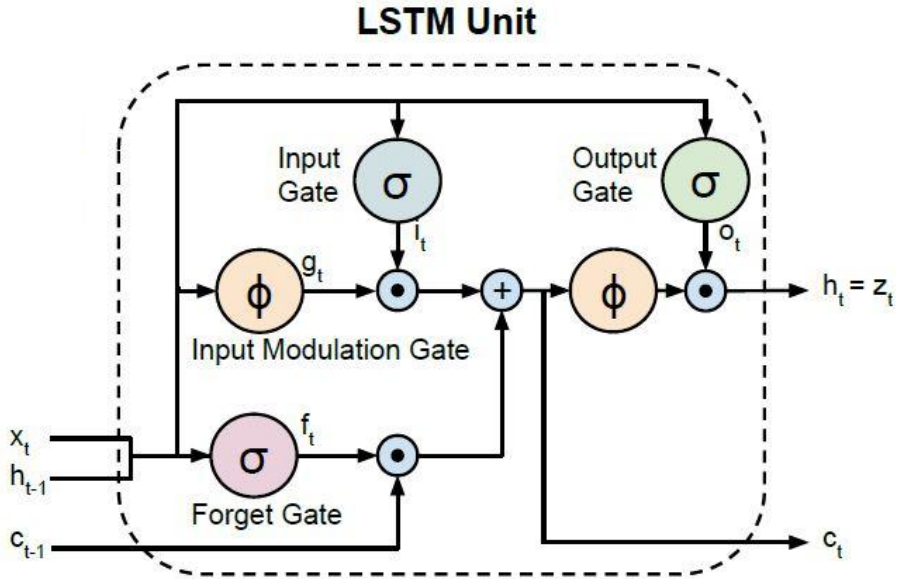
- Often found when training neural networks using backpropogation
- Hard to tune weights in earlier layers of network
- Causes RNNs to remember events that occurred a while ago



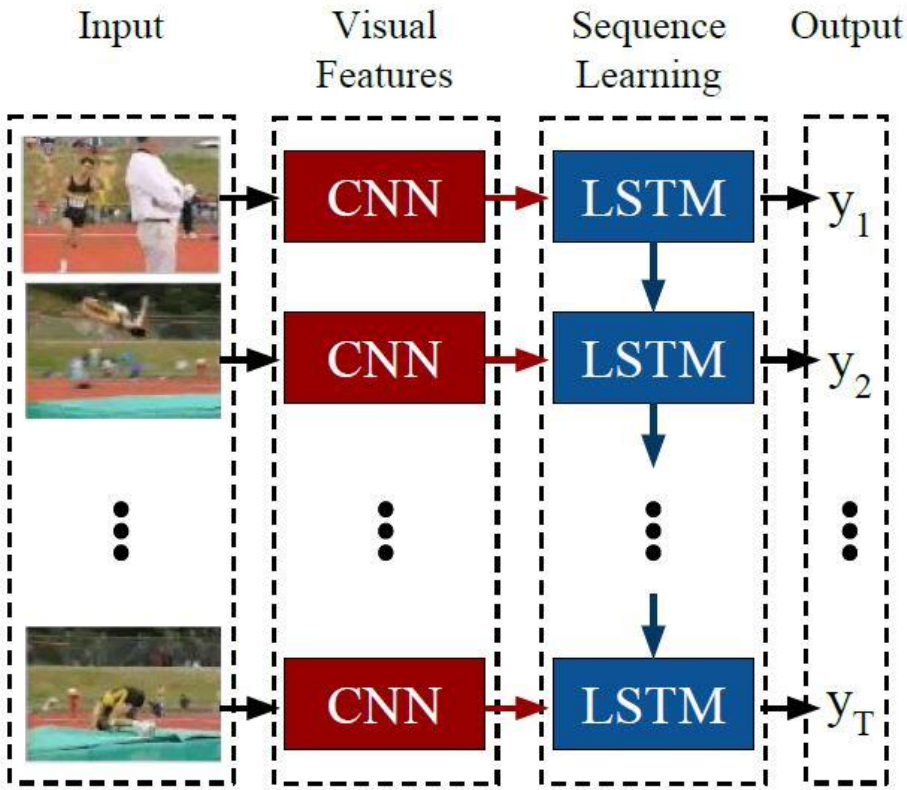


LSTMs - Long Term Memory

- Better for long term temporal data
- Have memory units that explicitly tell the network when to “forget” or update hidden states



Model


Input
Visual Features
Sequence Learning
Output
CNN
CNN
CNN
LSTM
LSTM
LSTM
y_1
y_2
y_T

Thanks