

# Master Thesis

A Library for Fast Kernel Expansions with Applications to  
Computer Vision and Deep Learning

H. C. Zarza

[zarza@cmu.edu](mailto:zarza@cmu.edu)

<http://www.andrew.cmu.edu/user/zarza/>

8th December 2014



# Outline

Motivation

C&Z Dataset

Fast Kernel Expansions: Randomized Features

McKernel

Applications

Conclusions

# Introduction

## Description

- Time period: 26th May 2014 - 5th December 2014
- Carnegie Mellon
- Location: Pittsburgh, Pennsylvania
- Office 8018, GATES HILLMAN Center
- School of Computer Science  
ML Department

# Motivation



# Motivation

- Explore the limitations of traditional Computer Vision
- Study novel techniques to accelerate learning in Large-scale Machine Learning: Fast Kernel Expansions
- Implement a library fast and easy-to-use
- Supplement with applications to Computer Vision and Deep Learning

# Traditional Computer Vision

- Building our own dataset: Exploiting Flickr
- Getting the labels: AMTurk
- Feature extraction: LBP Handcrafted Features around Landmark Facial Points
- Preprocessing step: Gamma Correction, Filter DoG and Contrast Equalization
- Classification: SVM Linear
- K-Fold Crossvalidation

# AMTurk

already have an account?  
[Sign in as a Worker](#) | [Requester](#)

**Your Account**   **HITS**   **Qualifications**

[Introduction](#) | [Dashboard](#) | [Status](#) | [Account Settings](#)

**Mechanical Turk is a marketplace for work.**  
We give businesses and developers access to an on-demand, scalable workforce.  
Workers select from thousands of tasks and work whenever it's convenient.

**326,752 HITS** available. [View them now.](#)

**Make Money**  
by working on HITs

HITs - Human Intelligence Tasks - are individual tasks that you work on. [Find HITs now.](#)

**As a Mechanical Turk Worker you:**

- Can work from home
- Choose your own work hours
- Get paid for doing good work

Find an interesting task → Work → Earn money

[Find HITs Now](#)

or [learn more about being a Worker](#)

**Get Results**  
from Mechanical Turk Workers

Ask workers to complete HITs - Human Intelligence Tasks - and get results using Mechanical Turk. [Get Started.](#)

**As a Mechanical Turk Requester you:**

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Fund your account → Load your tasks → Get results

[Get Started](#)

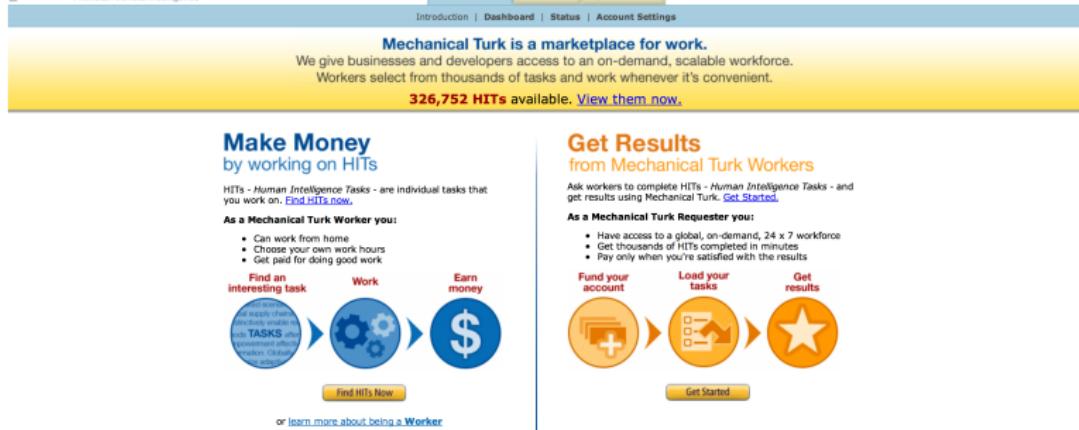


Figure: AMTurk

# Local Binary Patterns

Detect Facial Points using Supervised Descend [Xiong and De La Torre 2013] and then extract LBP Features around them

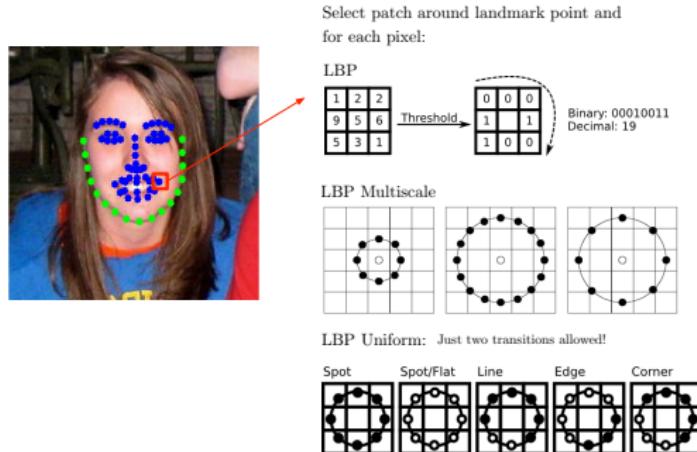


Figure: LBP

# Local Binary Patterns

## LBP Features

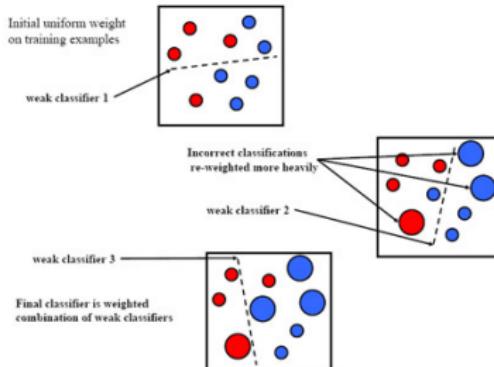
- LBP
- ULBP: less memory and computational time
- ULBP Multiscale: use of different radius to extract local and global information

**Performance improvement using a preprocessing step**

# Classifier

## 1. AdaBoost

AdaBoost

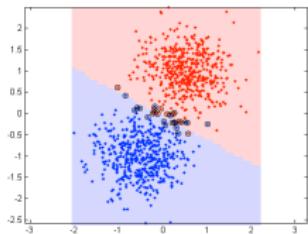


$$H(x) = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

## 2. SVM Linear and non-Linear

# Support Vector Machines

SVM Linear



SVM Kernel

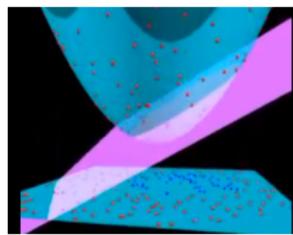
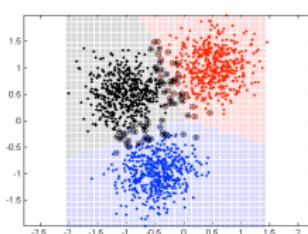
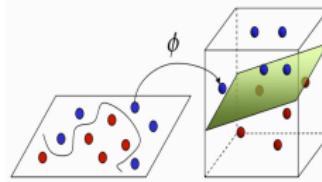
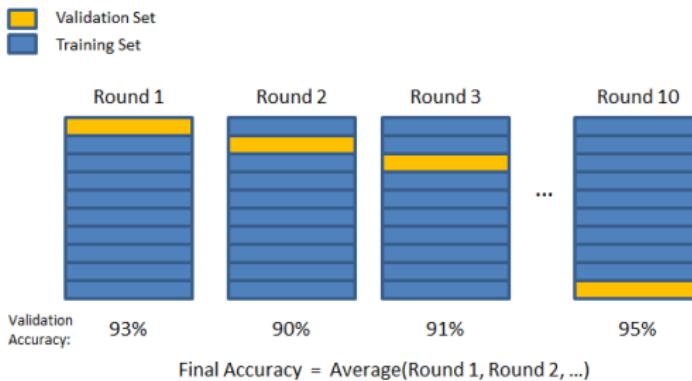


Figure: SVM

# Crossvalidation

Color Space, LBP parameters (radius, neighbors, patch size) and weak learners (AdaBoost)

## Crossvalidation



# Best Results

Color space	RGB	LUV	YCrCb	HSV
Accuracy (%)	77.4983	78.1971	78.2669	81.4116

Table: Color space K-Fold crossvalidation applied to classification of ethnicity

Test Explanation	Accuracy achieved (%)
ULBP, SVM Linear	77.71
ULBP Multiscale(3), SVM Linear	78.27
ULBP Multiscale(3), SVM Linear, HSV	81.42
ULBP Multiscale(3), SVM Linear, HSV, preprocessing	82.36
ULBP Multiscale(3), SVM Linear, HSV, optimized preprocessing	85.02

Table: Experimental Results System of Ethnicity

# Drawbacks and Solutions

## Drawbacks

- SVM non-linear entangles high cost in training step
- SVM not recommended for large datasets ( $> 50.000$  instances)

## Solutions

- Use Random Features to leverage learned training parameters
- Le et al propose Fastfood

# Fast Kernel Expansions: Randomized Features

In Random Kitchen Sinks instead of computing RBF GAUSSIAN Kernel

$$k(x, x') = \exp(-||x - x'||^2 / (2\sigma^2))$$

the method computes

$$k(x, x') = \exp(i[Zx]_c)$$

where  $z_c$  is drawn from a random distribution normal.

In Le et al  $Z$  is parametrized by  $V$  as

$$V := \frac{1}{\sigma\sqrt{d}} SHG\Pi HB$$

# McKernel

## Characteristics

- API following a design in Factory.
- Distributed-oriented version: Pseudo-random Numbers are generated using hash functions, no need to re-compute the matrices.
- Optimized library: cache-friendly code, unrolled loops, SIMD Intel Intrinsics for vectorized operations and in-place routines.

where

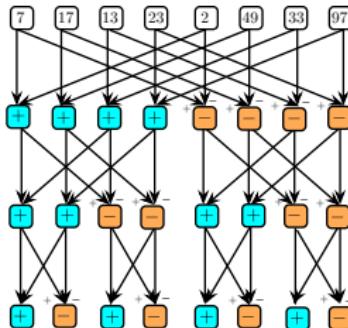
- $B$  entries 1 and  $-1$ .
- $H$  Walsh Hadamard. FWH maximizing cache hits and CPU performance. SIMD Intel Intrinsics.

Defining the  $1 \times 1$  Hadamard by the identity  $H_0 = 1$ , then  $\forall m > 0$ ,  $H_m$  is defined as:

$$H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix}$$

and for  $m > 1$  we have

$$H_m = H_1 \otimes H_{m-1}$$



# McKernel

- $\Pi$  permutation matrix using Fisher-Yates ( $O(n)$ ).
- $G$  entries following distribution Normal  $N(0, 1)$ .

Distributed-oriented version: BOX MULLER Transform

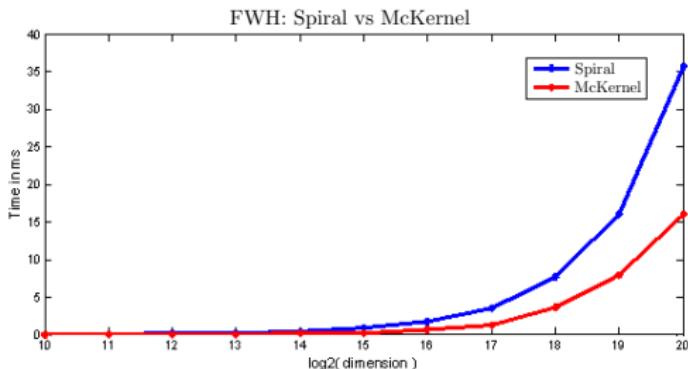
$$P_{cz} = (-2 \log h_1(c, z)/N)^{1/2} \cos(2\pi h_2(c, z)/N)$$

- $S$  entries random numbers chi with  $d$  degrees of freedom.  
Distributed version: Approximation by Wilson and Hilferty

$$\chi_d^2 = d \left( \sqrt{\frac{2}{9d}} z + \left(1 - \frac{2}{9d}\right) \right)^3$$

## Benchmarks

The experiments have been done using an Intel Core i5-4200 CPU @ 1.60 GHz machine. The results have been computed averaging the time performance of 300 random float vectors for each given length.

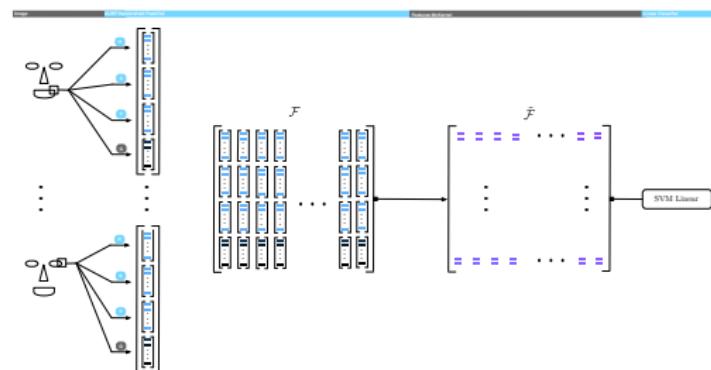


**Figure:** Comparison between Spiral and McKernel

# Application to Computer Vision

The feature map for McKernel is defined as:

$$\phi_c(x) = n^{-\frac{1}{2}} \exp(i[Vx]_c)$$

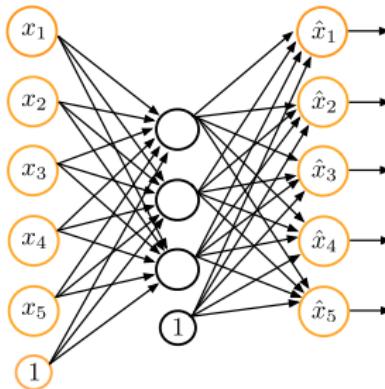


**Figure:** McKernel embedded in a System for Classification of Ethnicity.

# Application to Deep Learning

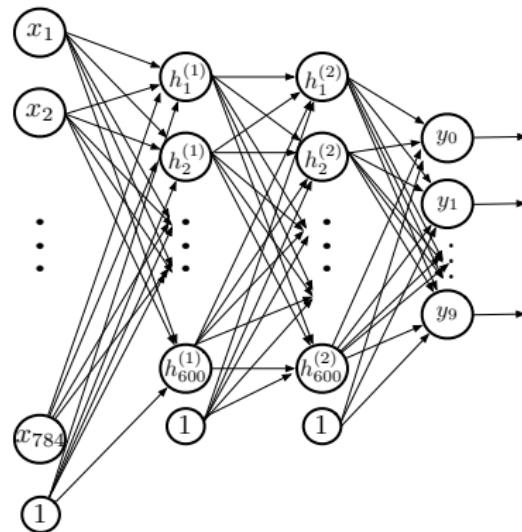
## Autoencoders

Extract the internal representation of the data by applying back propagation and setting  $y_{(z)} = x_{(z)}$ .



Stacked Autoencoders: Multiple layers of Sparse Autoencoders.

# Multi-layer Neural Network



**Figure:** Multi-layer Neural Network

# Applications to Deep Learning

## Code Highlights:

- MNIST Loading
- Implement function to compute the cost function and gradients for the sparse autoencoder, logistic regression and overall deep network.
- Implement the functions to check gradients are well computed.
- Train autoencoder layers and softmax regression.
- Fine-tune the network by backpropagation.

# Where does McKernel fit in?

We use McKernel as a non-linear mapping to the activation function.

## Results

MNIST Average Accuracy 96.31 %

3 % Improvement just by wiring McKernel.

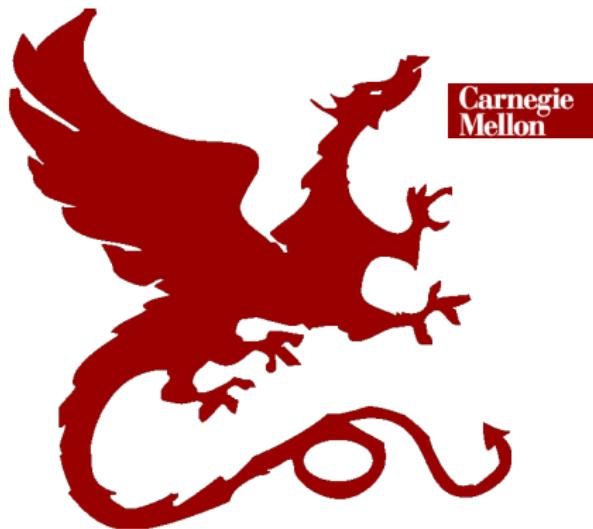
Additional gain by enlarging the number of Kernel Expansions.

# Conclusions

## Achievements

- C&Z Dataset
- SIMD FWH that performs better than current state-of-the-art libraries (Spiral)
- Fast Implementation of Approximate Kernel Expansions.  
Library McKernel
- McKernel embedded in a System for Estimation of Ethnicity
- McKernel wired in Deep Learning

Thank you



DE ZARZA I CUBERO Irene

Thank you

Warm thank you to all the people at the ML Department, Robotics and Carnegie Mellon that made this possible.