Machine Learning for Facial Expression Recognition: A review

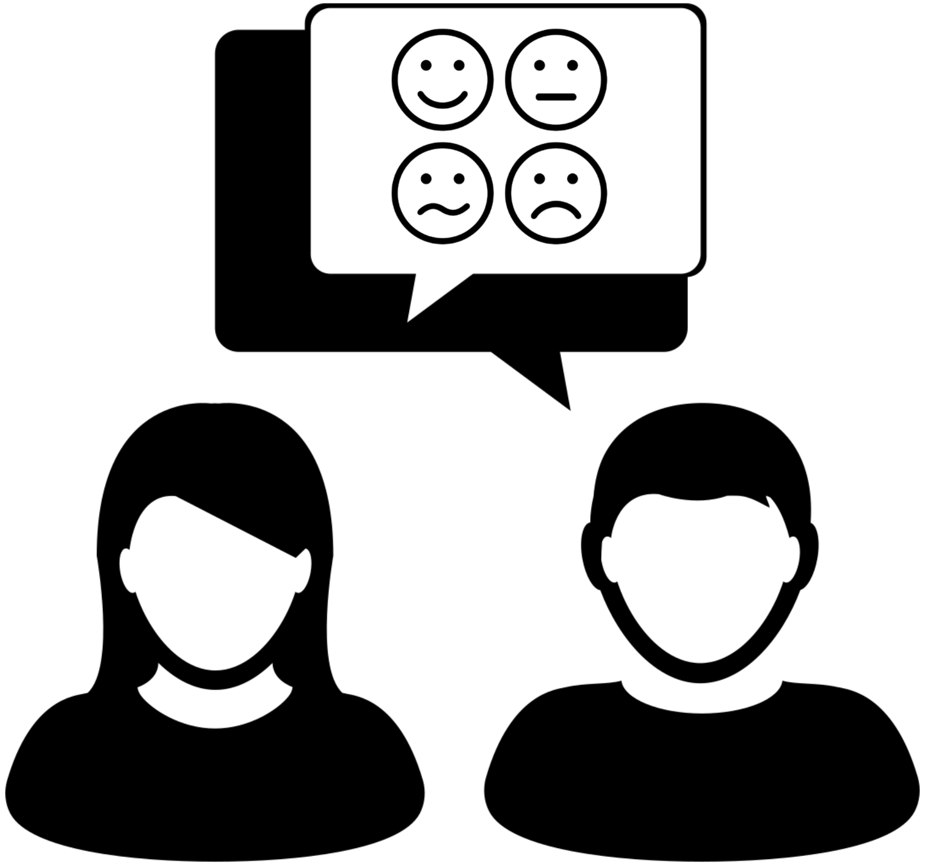
Course: Problems in Machine Learning // ECE-551 // Fall 2022

Prepared by: Silas Curfman

Instructor: Dr. Marios Pattichis

Outline

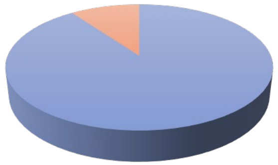
I. Motivation
II. Background and History
III. State of the Art
IV. Emerging Methods
V. Demonstration
VI. Future Modifications
VII. Conclusion
VIII. Question & Answer



Motivation

- What is Machine Learning for 'Facial Expression Recognition'?
  - Computer based
  - Digital Images / Video
  - Human Faces
  - Input [Expression]
  - Output [Emotion]

- Why do we care?
  - Communication = 10%(verb) + 90%(nonverbal)



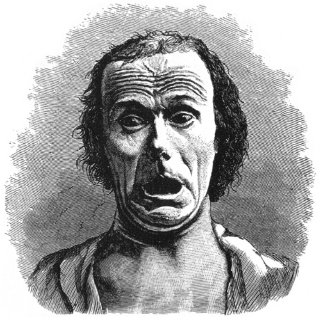
  - 75% nonverbal misinterpreted
  - Universality, transcendant problem.
  - ROI (little more effort, big payoff)
  - Applies everywhere, every field

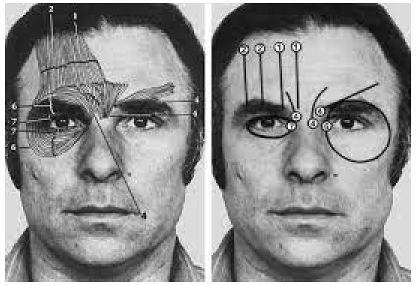
Application / Use Cases

- Societal
  - Public relations
  - Politics
  - Customer / Client
  - Public safetly / Law enforcement
  - Foreign relations, diplomacy
- Interpersonal
  - Healthcare. Doctor / Patient
  - Behavioral sciences
  - Caregiving for nonverbal or semi-verbal individuals
    - Strokes, dementia, schizophrenia, autism, etc.
  - Hearing / Speech impairments
  - Language translation / interpretation

History

- Pre-Digital Revolution / Analog
  - Emotions ‘coded’ into ancient art. Sculptures, masks, paintings.
  - 1872 Darwin publishes ‘The Expression of the Emotions in Man and Animals’.
  - 1957 Paul Eckman publishes methods for measuring nonverbal comms. Deception detection
  - 1989 Gottman & Krokoff introduce Specific Affect Coding System (SPAFF). Marriage & Family counseling, improve communication.





AUs 4+5, with lip press, associated with Anger, Criticism, Contempt

Unilateral AU14, associated with Contempt

Unilateral AU14 with eye roll, associated with Contempt

AUs 4+10, associated with Disgust and Contempt

AU 2 (“the horns”), associated with Domineering

AU 2 (“the horns”) with head forward, associated with Domineering

Present Age

- Post-Digital Revolution
  - **1964** Bledsoe experiments with computers detecting faces
  - **1970's** Harmon & Lesk extend detection to recognition.
  - **1980's** Sirovich & Kirby apply linear algebra, 'Eigenface'
  - **1989** LeCun publishes backpropogation method (Deep Learning)
  - **1990's** DARPA & NIST introduce Face Recognition Technology (FERET)
- **2000's** Social Media explosion, sudden availability of large datasets.
- **2015** Google releases Tensorflow as open source technology.
- **2016** PyTorch machine learning framework released.
- …. **Present**: Open source expression based datasets freely available, both shallow and deep FER methods established, increased access to

Emerging Concepts / Models

- Deep Learning Long Short-Term Memory (LSTM) + Reinforcement Learning
- Affordable access to HPC & Cloud Computing
- Increased availability of large datasets

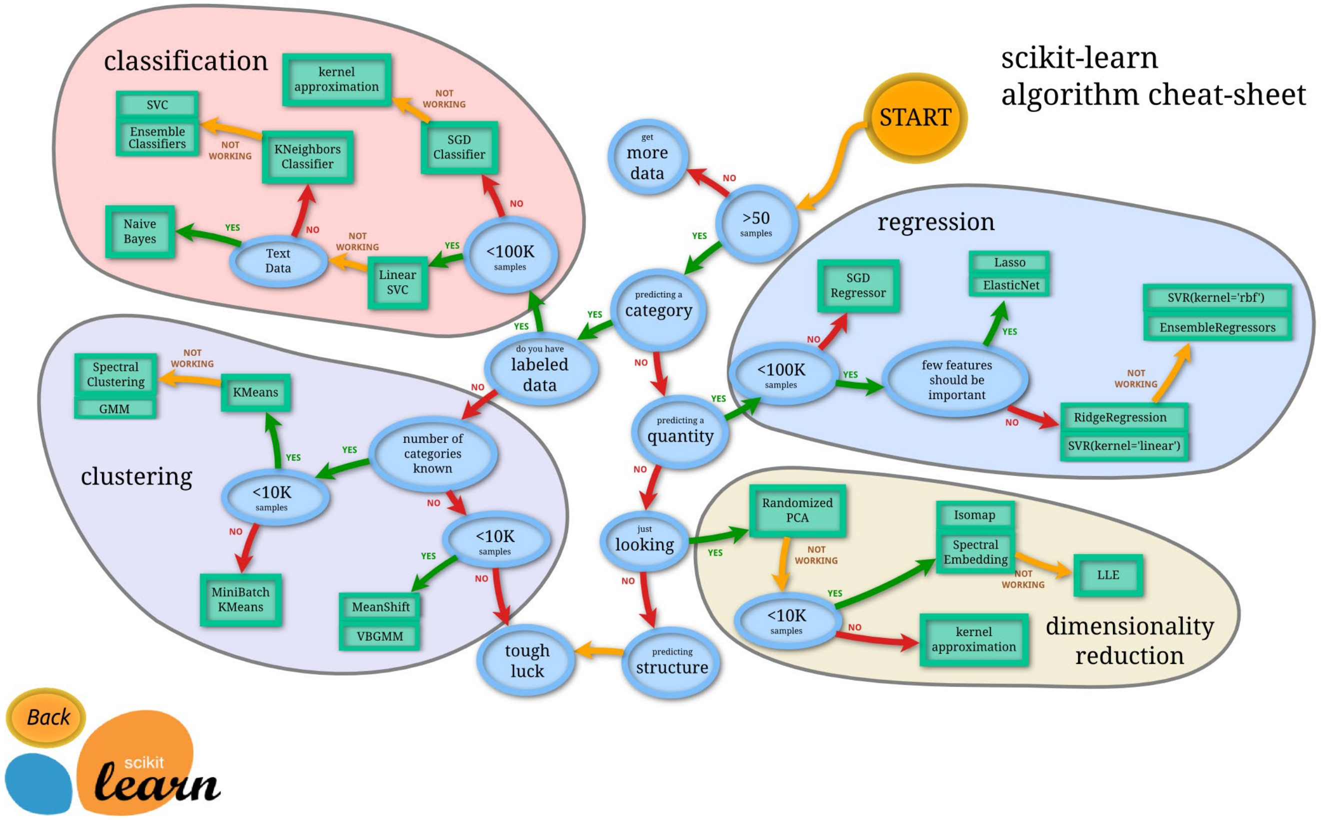
30,000ft view

- Machine Learning is not ....
  - Exact
  - One size fits all
  - Deterministic
  - Proofs and axioms
- *"DETERMINISTIC"*

- Machine Learning is ...
  - Approximate
  - Multiple tools can do same job
  - Generalistic
  - Statistical patterns
- *"STOCHASTIC"*

Where to start?

- Define the problem. Prediction
- Predict what? Quality or Quantity?
- How many features, classes, labels, outputs are we interested in?
- With what resources, datasets, hardware?
- At what cost. Time == $$$



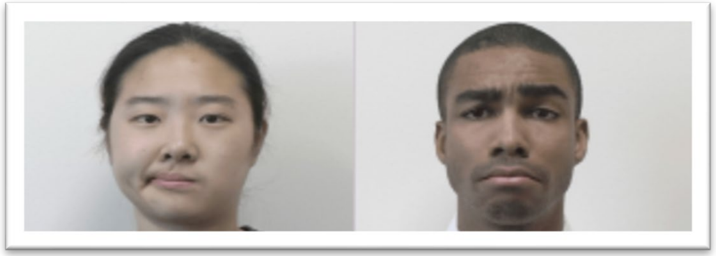
ELECTRICAL
& COMPUTER
ENGINEERING

3 FER Datasets

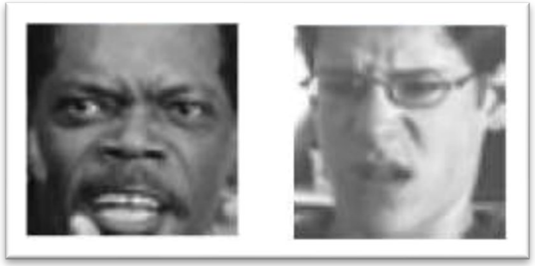
(7 labeled emotions)

- JAFFE
  - Small dataset, 213 labeled images
  - Preprocessed: 8-bit greyscale, 256 x 256.
  - Highly regularized
    - Uniform age, ethnicity, sex, & pose
  - Very low entropy

- CK+
  - Moderate complexity, 5,876 labeled images
  - RGB color images, 640 x 480
  - Generalized
    - Multipe ages, ethnicities, sex
  - Medium entropy

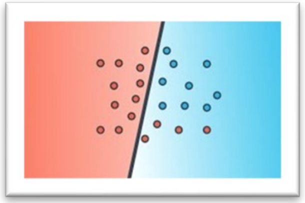


- FER-2013
  - Large dataset, 30,000 labeled images
  - Greyscale images, 48 x 48
  - Generalized
    - Multipe ages, ethnicities, sex
  - High entropy

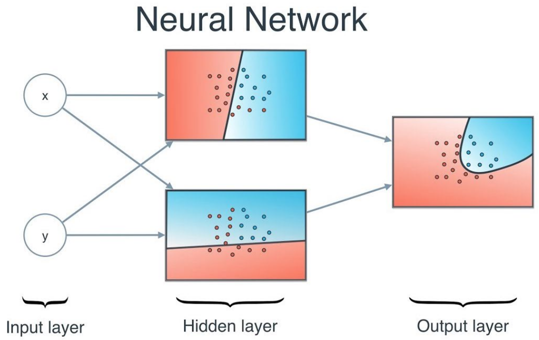


3 FER Models

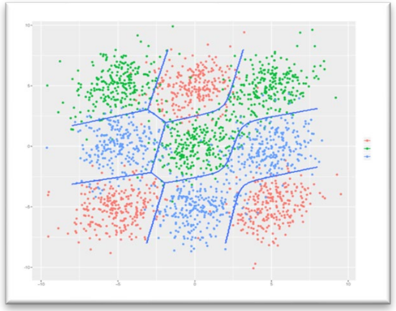
- SVM's
  - Easy to construct
  - Small datasets = OK
  - Best for binary classification but can be multi-class
  - Single layer



- MLP'S
  - Moderate complexity
  - Getting into neural nets
  - Combining multiple classifiers for better output
  - Multiple classes = OK
  - Multiple layers



- CNN's
  - High complexity
  - Best with large dataset
  - Inputs tensors instead of vectors, retains special relations between pixels
  - Nonlinear = OK
  - Multiple hidden layers



Data Preprocessing

ELECTRICAL & COMPUTER ENGINEERING

Load it …

```
# Load the dataset from the disk

def get_label_from_filename(filename):
  """ Given a filename of the format 'NM.NE2.93.tiff', return the label 'NE'."""
  index = filename.find('.')
  return filename[index+1:index+3]
```

Label it…

```
emotion_to_int =  {"AN":0, "DI":1, "FE":2, "NE":3, "SA":4, "SU":5, "HA":6}
int_to_emotion =  {0:"AN", 1:"DI", 2:"FE", 3:"NE", 4:"SA", 5:"SU", 6:"HA"}
emotion_list = emotion_to_int.keys()

img_data_list = []
labels_list = []
```

Flatten it…

```
img_data = np.array(img_data_list)
img_data = img_data.astype('float32')
img_data = img_data/255 # Normalize between [0-1]
img_data = img_data.reshape((len(img_data), -1)) # Flatten the images
labels = np.array(labels_list)
```

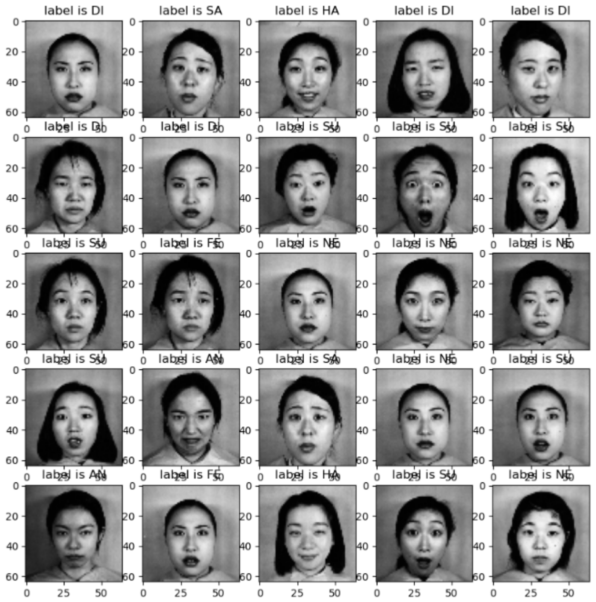
Split it…

```
# Split the data into train and test set
train_size = int(num_images*0.8) # reserve 80% for training, 20% for testing
train_images = img_data[0:train_size]
train_labels = labels[0:train_size]
test_images = img_data[train_size:]
test_labels = labels[train_size:]
```

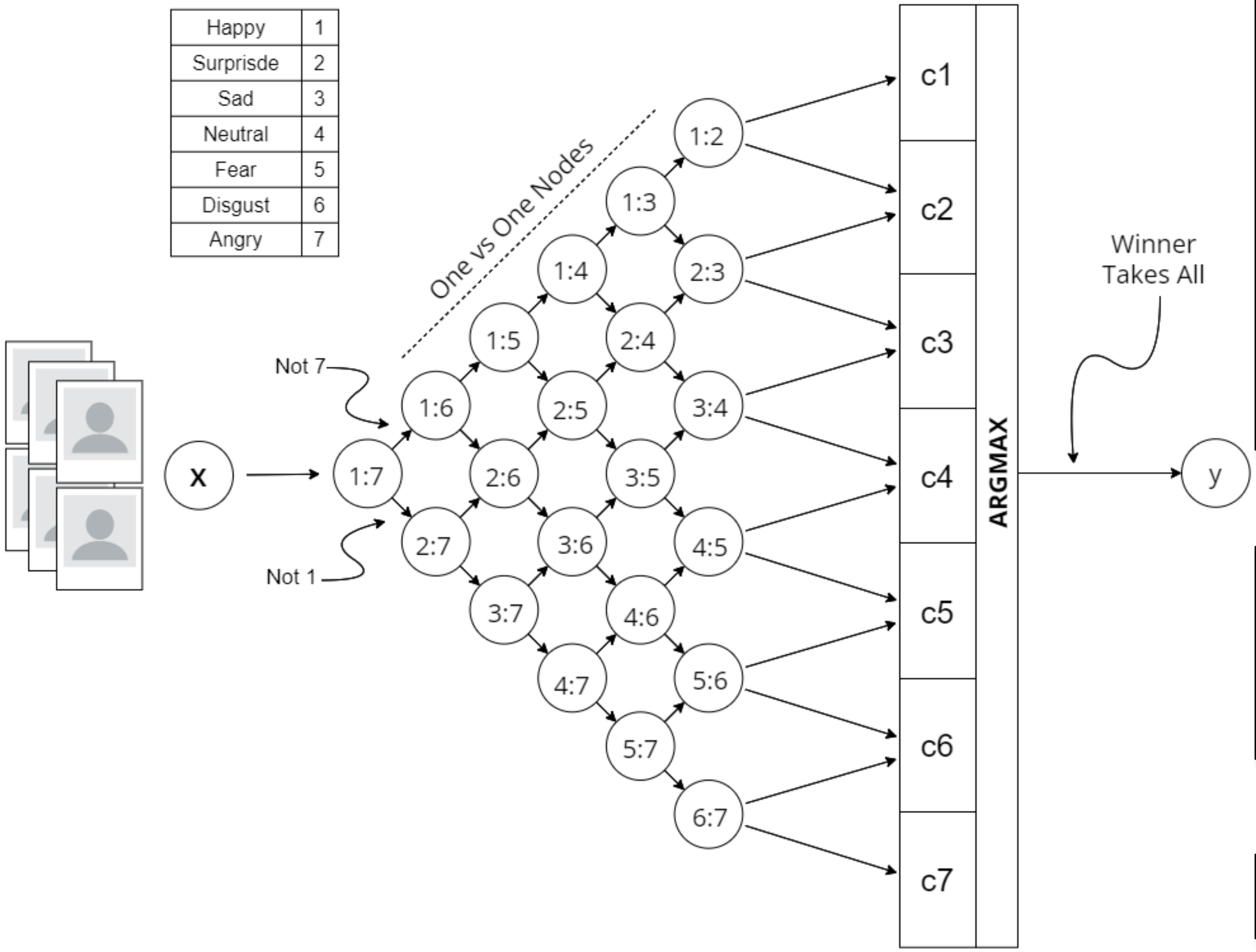
Display it…

```
# Create an NxN display of samples
N = 5
fig, axs = plt.subplots(N, N)

for i in range(5):
  for j in range(5):
    index = random.randint(0, 213-1) # pick a random index
    img = img_data[index]
    img = img.reshape(cols, rows, channels)
    label = labels[index]
    axs[i, j].imshow(img)
    axs[i, j].set_title("label is " + int_to_emotion[label])
```



SVM FER Demo (JAFFE)



SVM Setup...

```
from sklearn import svm, metrics

# Create a classifier: a support vector classifier
# This is with RBF kernel
classifier = svm.SVC(gamma=0.001)

print("classifier: ", classifier)
# C: # OVR: One-versus-rest (alternative: ovo -- One-versus-one)
# Kernel (RBF): Radial Basis Functions
# Probability (False): Estimate the probability for class membership from scores
# class_weight (None): Give more weight to some classess
# coef0: Constant r in the kernel definition (see above)
```

SVM Train & Predict...

```
# Train the SVM model on the training data
classifier.fit(train_images, train_labels)

# Now predict on the test data
predicted = classifier.predict(test_images)
expected = test_labels
```

SVM Measure Performance...

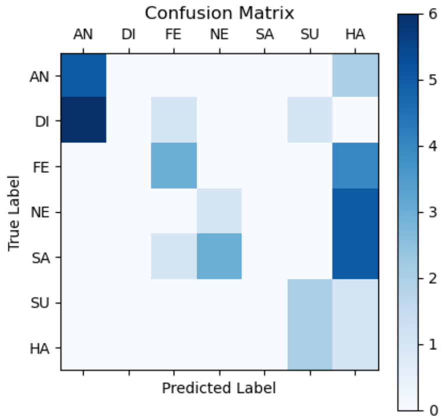
```
print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(expected, predicted)))
```

SVM FER Demo (JAFFE) Results

- Initial Configuration
  - C: 1.0
  - Kernel: rbf
  - Gamma: 0.001
  - Multiclass: one vs one

- Initial Results

Score	SVM (0)
Precision	0.86
Recall	0.81
F1	0.80

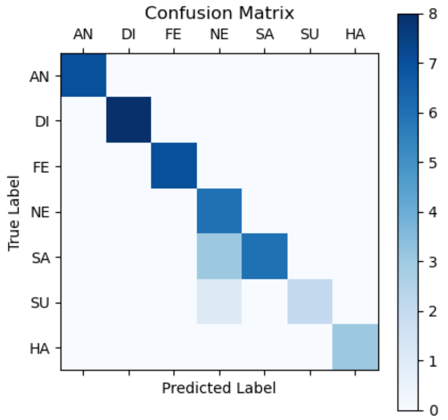


- Gridsearch

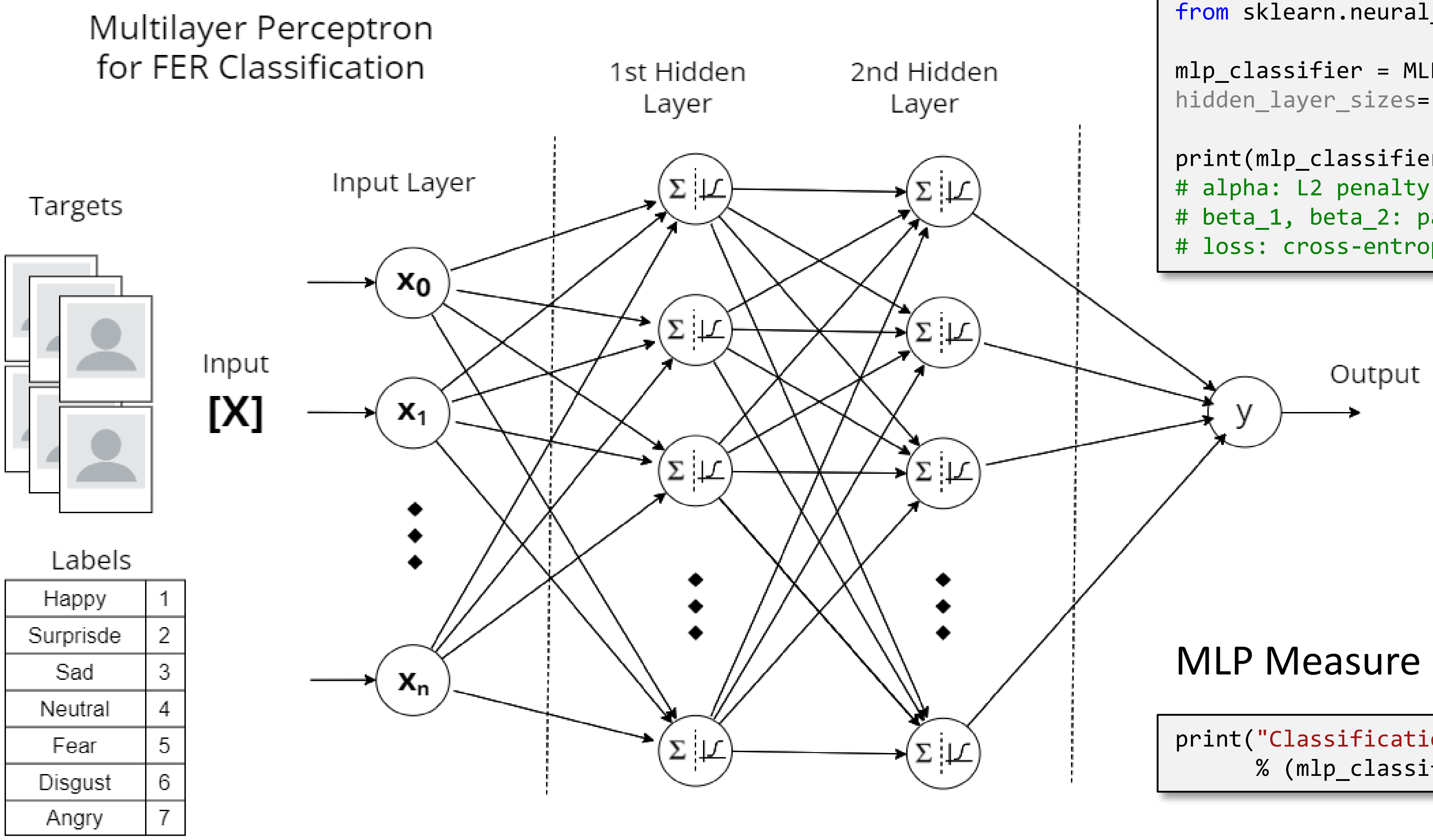
HyperParam	MLP
C	1.0e(-7)
Kernel	rbf
gamma	1.0e(-6)

- Optimized

Score	SVM (0)
Precision	0.94
Recall	0.91
F1	0.91



MLP FER Demo (JAFFE)



MLP Setup…

```
from sklearn.neural_network import MLPClassifier

mlp_classifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(1000, 200), random_state=1, verbose=True)

print(mlp_classifier)
# alpha: L2 penalty (regularization term) parameter.
# beta_1, beta_2: parameters for first-order and second-order moments of Adam
# loss: cross-entropy loss.
```

MLP Train & Predict…

```
# We learn the SVM model on the training data
mlp_classifier.fit(train_images, train_labels)

# Now predict on the test data
predicted = mlp_classifier.predict(test_images)
expected = test_labels
```

MLP Measure Performance…

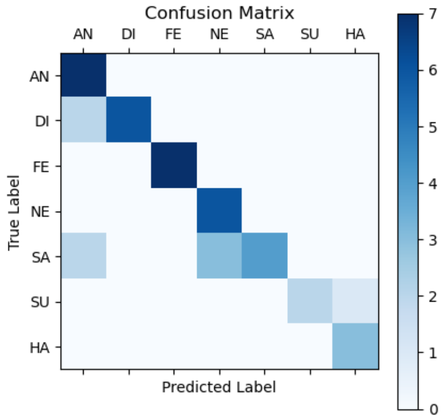
```
print("Classification report for classifier %s:\n%s\n"
      % (mlp_classifier, metrics.classification_report(expected, predicted)))
```

MLP FER Demo (JAFFE) Results

- Initial Configuration
  - Solver: lbfgs
  - Alpha: 0.0001
  - Hidden Layers / Nodes: H1:1000N, H2:200N
  - Split: 80% Train, 10% Test, 10% Validate

- Initial Results

Score	SVM (0)
Precision	0.86
Recall	0.81
F1	0.80

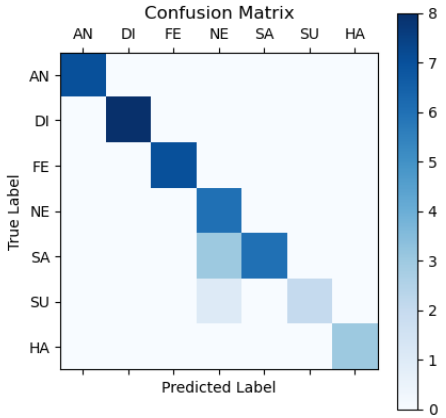


- Gridsearch

HyperParam	MLP
Solver	SGC
Alpha	0.0001
Layers	H1:200N H2:200N H3:200N H4:200N H5:200N H6:200N

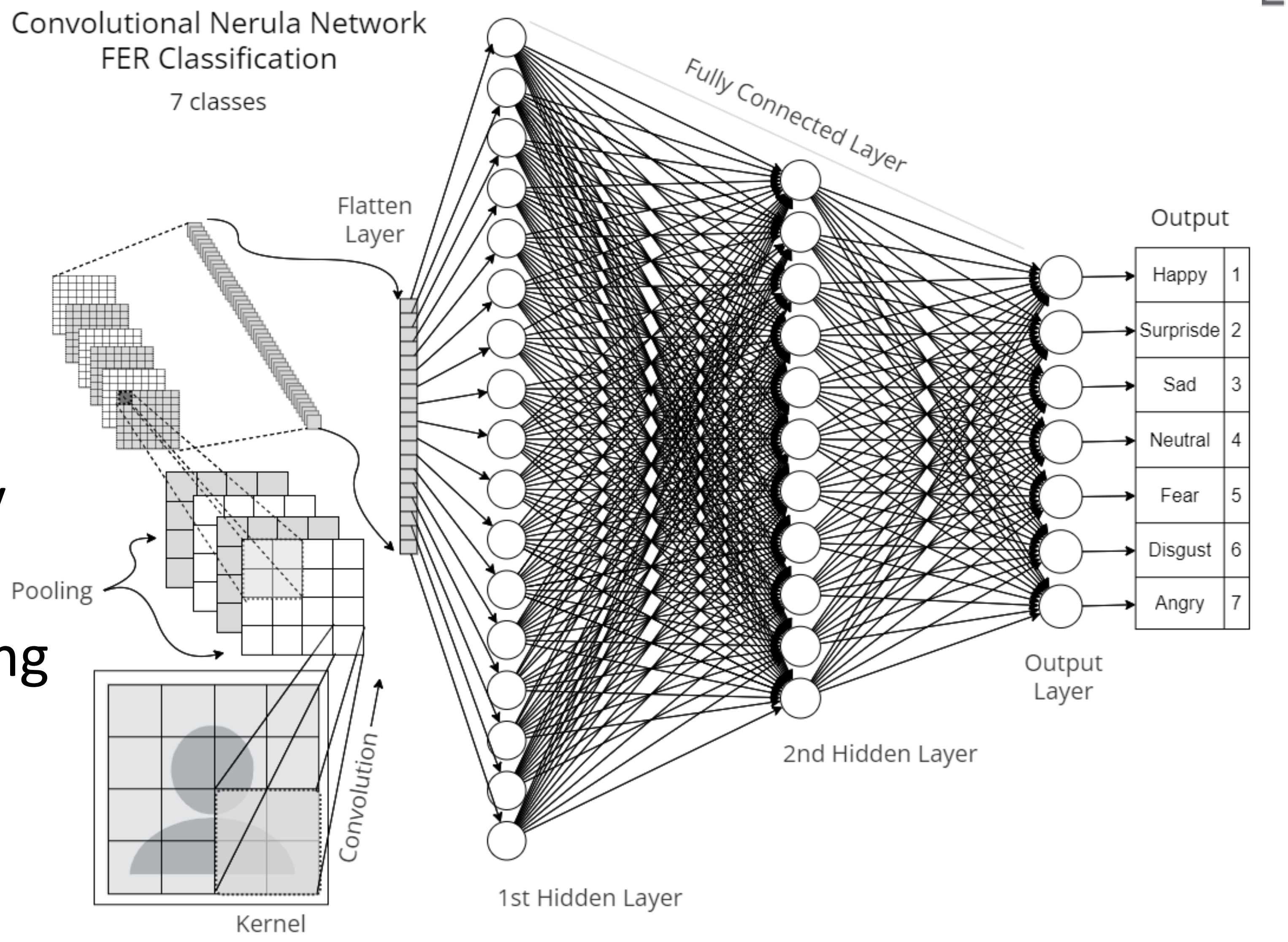
- Final Results

Score	SVM (0)
Precision	0.94
Recall	0.91
F1	0.91



Proposed Modifications

- Deep learning, CNN
- Kernel decomposition of images to retain special relationships to better handle randomly framed images.
- Host via cloud computing platform for real time access



Conclusion / Remarks / References

- SVM / MLP Python Notebook
  https://github.com/blueCollarSysadmin/ECE551/blob/804b4f886939938ff6c1686463e1e70e07fa77f2/ece551ferSvmMlp.ipynb
- CNN Demo Notebook (FER-2013 data set)
  https://github.com/blueCollarSysadmin/ECE551/blob/804b4f886939938ff6c1686463e1e70e07fa77f2/ece551ferCnnTflow.ipynb
- JAFFE Dataset
  https://zenodo.org/record/3451524#.Y04IXnbMJhE
- CK+ Datset
  https://ieeexplore.ieee.org/document/5543262/references#references
- FER-2013 Dataset
  https://www.kaggle.com/datasets/msambare/fer2013
- Written version of this report:
  https://github.com/blueCollarSysadmin/ECE551/blob/d751edfbeea0ed1d2606f99995499021984245c7/unmEce551_machineLearningForFER_silasCurfman_fall_2022.pdf