Data 240

Facial Expression Recognition Applications Performance Evaluation

Team 2
Mavis Wang, Gaven Zheng, Xiaocen Xie, Coco Yu, Yan Tang

Intro

- Motivation
- Data Collection
- General Approach
- Preprocessing
- Training 3 models
- Ensemble Experience and Result
- Conclusion/ future work

Motivation

- Street crime is always a major concern in big cities

- Criminal behaviour has been connected to a lack of emotion awareness, notably for fear, rage, and other emotions

- Proposed Solution: using the deep Convolutional Neural Network (CNN) technique to detect facial expressions
  3 models: ResNet50V2, Mini-VGG, InceptionV3

Data Collection

Facial Expression Recognition 2013 Dataset(FER2013)

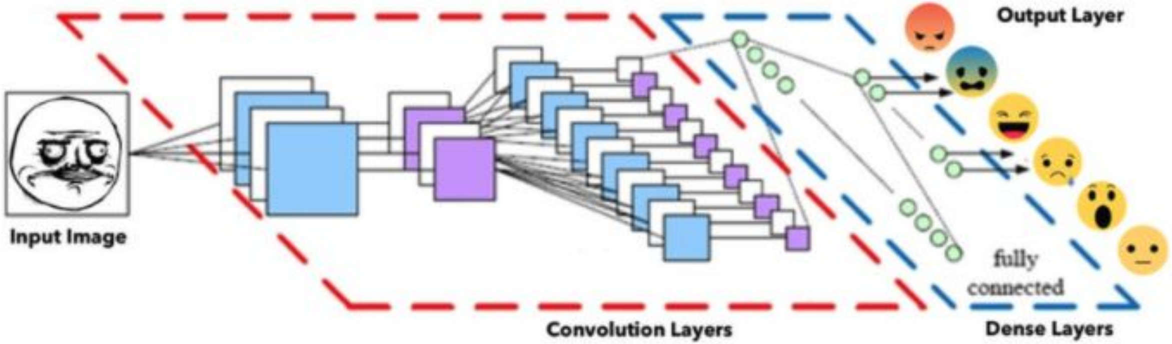
- Contains 30,000 facial RGB images of different expressions
- The data consists of 48x48 pixel grayscale images of faces
- Seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)

General Approach

- Use FER2013 dataset. The training set consists of 28,709 examples and the public test set consists of 3,589 examples.

- Train 3 models: ResNet50V2, Mini-VGG, InceptionV3

- Evaluate the performance of each model



Preprocessing

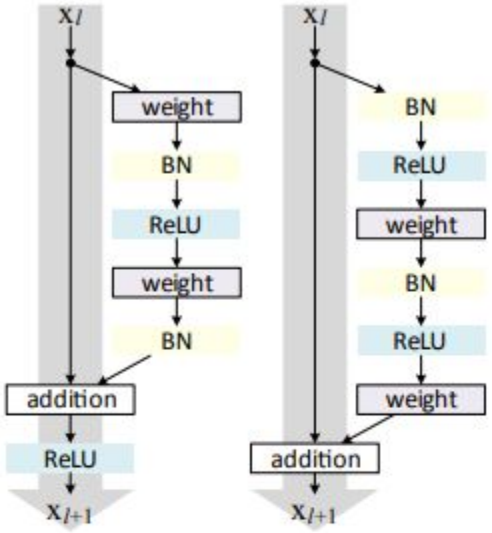
- Normalization
  - Rescal by 1 / 225.0
- Channel Repetition
  - Pre-trained models support only 3-channel images
  - Repeat the single channel 3 times (gray scale)
- Image Augmentation
  - Augmentation during training (rotation, h/v flip, etc.)

Models

- ResNet50V2
- Mini-VGG
- Ensemble

ResNet50V2

- Residual Block V1
    - Adds the second non-linearity after the addition.
    - Performs the convolution before BN and ReLU.
- Residual Block V2
    - Remove the linearity after addition
    - Applies BM and ReLU to the input before the convolution operation



ResNet50V2

- Transfer Learning Using ResNet50V2
  - Freeze ResNet
  - Adding additional FC layer

```
model = tf.keras.Sequential([
    Input(shape=(IMAGE_SIZE[0],IMAGE_SIZE[1],3)),
    data_augmentation,
    res_net_v2,
    layers.GlobalAveragePooling2D(),
    tf.keras.layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(1024, activation="elu", kernel_regularizer = tf.keras.regularizers.l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    layers.Dropout(0.4),
    layers.Dense(512, activation="elu"),
    tf.keras.layers.BatchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(128, activation="elu"),
    layers.Dropout(0.2),
    layers.Dense(7, activation="softmax")
])
```

ResNet50V2

- Freeze and train - 50 epoches
- Unfreeze the ResNet - 50 epoches
    - Trying a smaller learning rate
- Assign Class Weights -20 epoches
    - Trying even smaller learning rate

```
model.layers[1].trainable = True
```

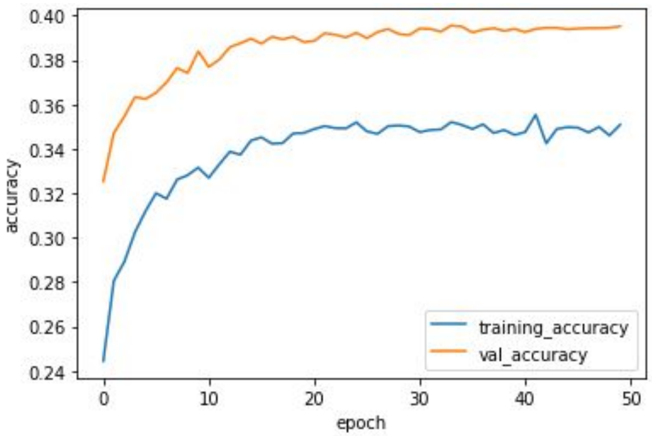
```
{0: 3196, 1: 349, 2: 3278, 3: 5772,
4: 3972, 5: 3864, 6: 2537}
```

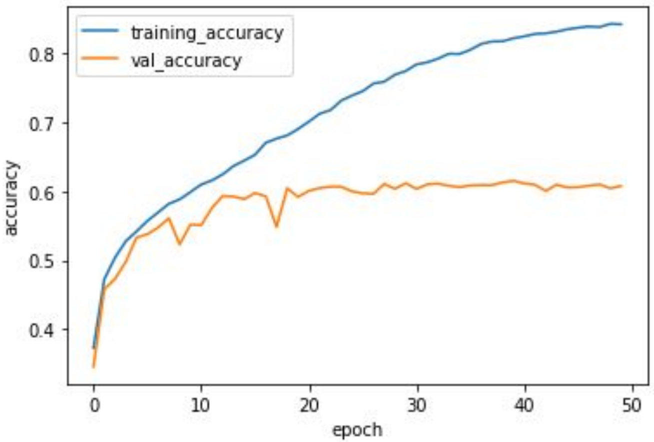
```
{0: 1.0266404434114071,
 1: 9.401555464592715,
 2: 1.0009587727708533,
 3: 0.5684585684585685,
 4: 0.826068191627104,
 5: 0.8491570541259982,
 6: 1.2933160650937552}
```

Class Weights

ResNet50V2 Training

Left: Freeze ResNet; lr = 0.001
Right: Unfreeze ResNet; lr = 0.0005



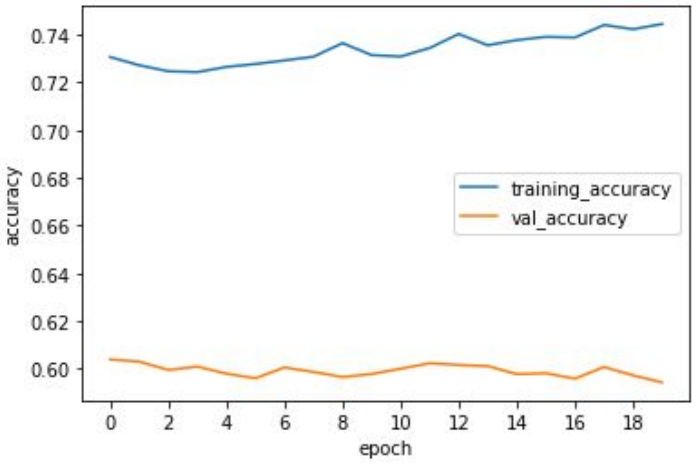


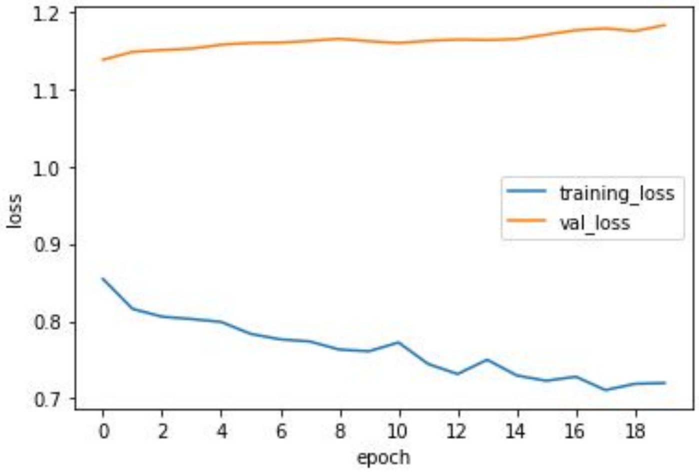
Class Weighting

- Assign weights to each class
  - `{0: 3196, 1: 349, 2: 3278, 3: 5772, 4: 3972, 5: 3864, 6: 2537}`
  - lr=0.00005

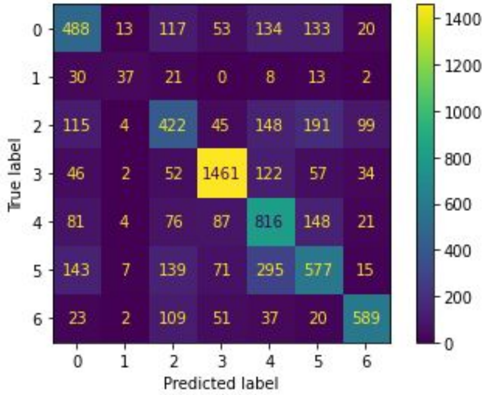
```
{0: 1.0266404434114071,
 1: 9.401555464592715,
 2: 1.0009587727708533,
 3: 0.5684585684585685,
 4: 0.826068191627104,
 5: 0.8491570541259982,
 6: 1.2933160650937552}
```

Class Weights

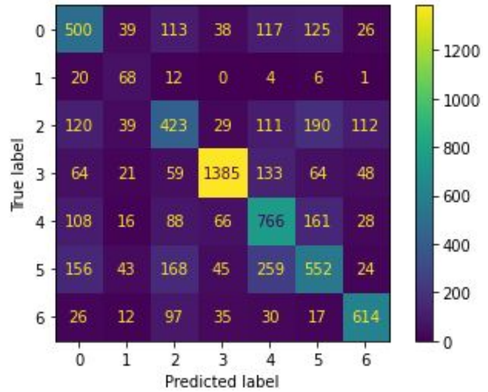




ResNet50V2 Result



	precision	recall	f1-score	support
0	0.5270	0.5094	0.5180	958
1	0.5362	0.3333	0.4111	111
2	0.4509	0.4121	0.4306	1024
3	0.8264	0.8236	0.8250	1774
4	0.5231	0.6618	0.5843	1233
5	0.5066	0.4627	0.4837	1247
6	0.7551	0.7088	0.7312	831
accuracy			0.6116	7178
macro avg	0.5893	0.5588	0.5691	7178
weighted avg	0.6125	0.6116	0.6099	7178

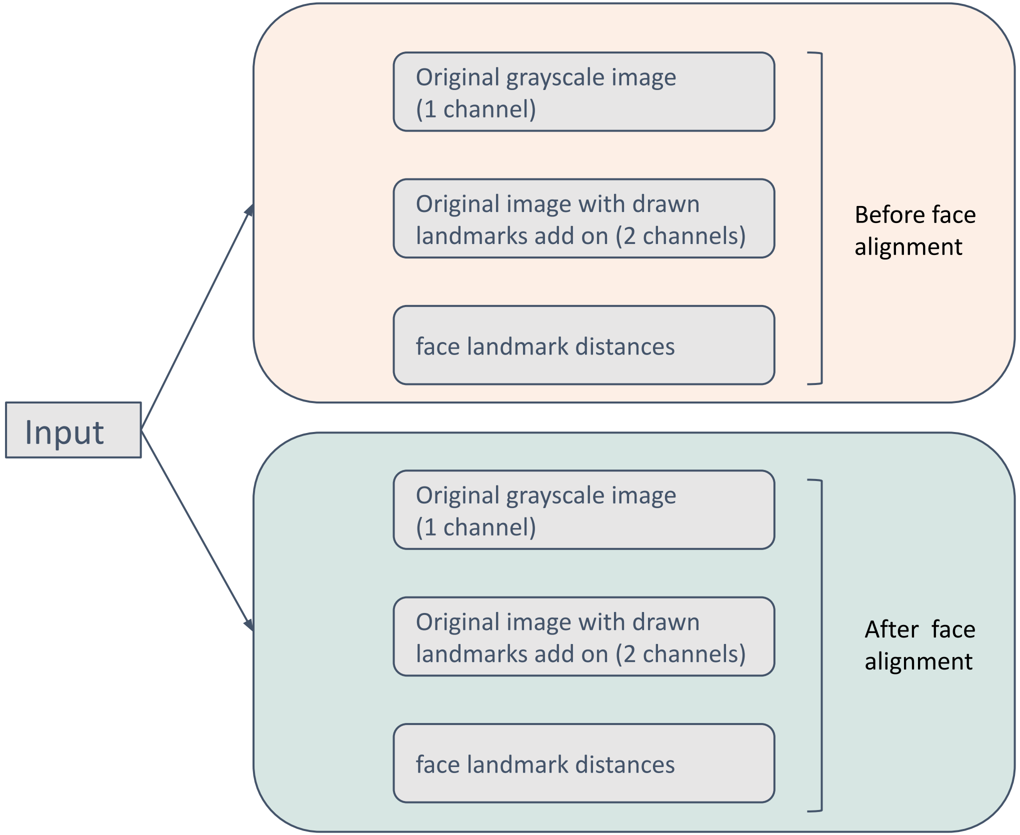


	precision	recall	f1-score	support
0	0.5030	0.5219	0.5123	958
1	0.2857	0.6126	0.3897	111
2	0.4406	0.4131	0.4264	1024
3	0.8667	0.7807	0.8215	1774
4	0.5394	0.6212	0.5775	1233
5	0.4951	0.4427	0.4674	1247
6	0.7198	0.7389	0.7292	831
accuracy			0.6002	7178
macro avg	0.5501	0.5902	0.5606	7178
weighted avg	0.6106	0.6002	0.6031	7178

Mini-VGG

Feature Extraction And Transformation Methods We Tried :

1. Extract face landmarks using dlib (48 x 48 features $\rightarrow$ 72 features)
2. Extract landmark distances from face landmarks (72 features $\rightarrow$ 8)
3. Add an empty image with drawn face landmarks
4. Face alignment



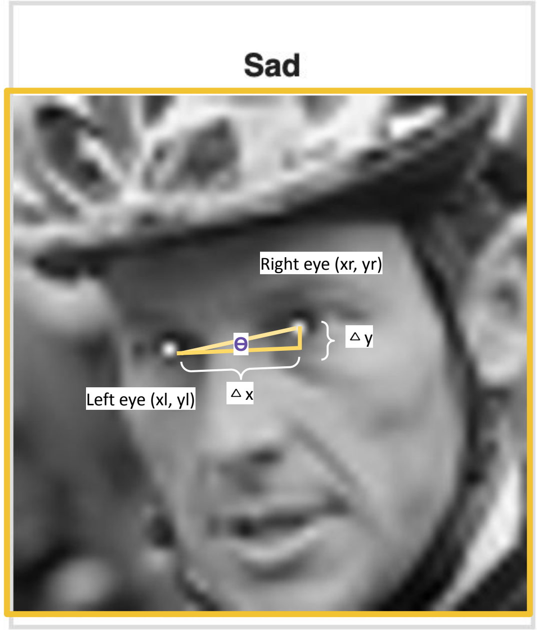
Mini-VGG

Extract face landmarks using dlib (48 x 48 features $\rightarrow$ 72 features)


```
('chin', [(10, 25), (10, 29), (12, 33), (13, 36), (15, 40), (17, 43), (19, 46), (22, 48), (26, 48), (30, 48), (34, 45), (38, 42), (40, 38), (42, 33), (42, 28), (41, 23), (40, 18)])
('left_eyebrow', [(9, 21), (10, 19), (11, 19), (13, 19), (15, 20)])
('right_eyebrow', [(20, 18), (23, 17), (25, 16), (28, 15), (31, 16)])
('nose_bridge', [(18, 23), (18, 26), (18, 29), (18, 32)])
('nose_tip', [(17, 34), (19, 35), (20, 35), (22, 34), (24, 33)])
('left_eye', [(12, 24), (13, 22), (15, 22), (17, 24), (15, 24), (13, 25)])
('right_eye', [(24, 22), (25, 20), (27, 20), (30, 20), (28, 22), (26, 22)])
('top_lip', [(18, 40), (19, 39), (20, 38), (22, 39), (24, 38), (27, 38), (30, 38), (29, 38), (24, 39), (22, 40), (20, 40), (19, 40)])
('bottom_lip', [(30, 38), (28, 41), (25, 43), (23, 43), (21, 43), (20, 42), (18, 40), (19, 40), (21, 41), (22, 41), (24, 41), (29, 38)])
```

Mini-VGG

Face alignment



Face Alignment

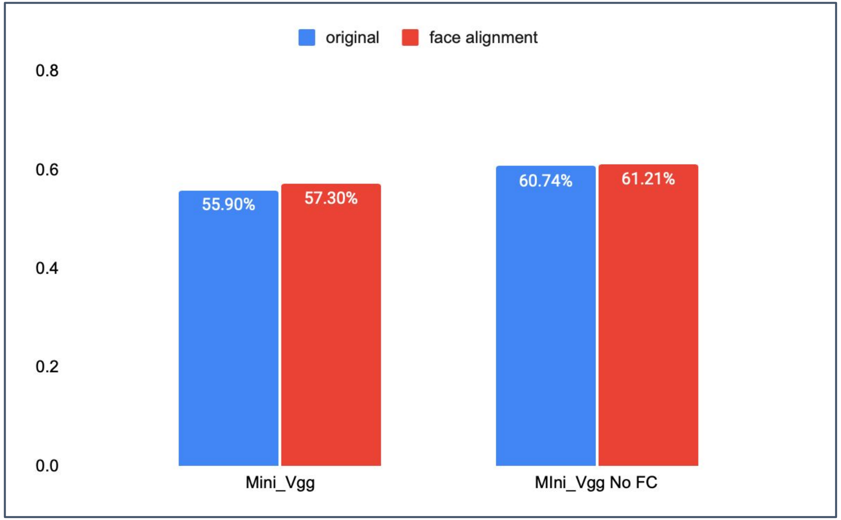
$$\Delta x = x_r - x_l$$

$$\Delta y = y_r - y_l$$

$$\theta = \arctan\frac{\Delta y}{\Delta x}$$

Mini-VGG

Mini_Vgg	Mini_Vgg No FC
Input	Input
3 x 3 Conv, 64	3 x 3 Conv, 64
3 x 3 Conv, 64	3 x 3 Conv, 64
Pool	Pool
3 x 3 Conv, 128	3 x 3 Conv, 128
3 x 3 Conv, 128	3 x 3 Conv, 128
Pool	Pool
3 x 3 Conv, 256	3 x 3 Conv, 256
3 x 3 Conv, 256	3 x 3 Conv, 256
Pool	1 x 1 Conv, 128
FC 512	1 x 1 Conv, 7
FC 7	Global AvgPool
Softmax	Softmax



Model variants:

- activation function: Relu
- All convolution layer using strip = 1, padding = 1
- Loss Function: CrossEntropyLoss()
- Optimizer: Adam()
- Batchsize = 100

Mini-VGG

Extract landmark distances from face landmarks (72 features → 8)

- Performance is not good

```
[0.4850712500726659,
 12.165525060596439,
 2.0,
 0.0,
 -0.24497866312686414,
 3,
 2,
 1.3518824678560455]
```

```
"""
distance feature
0. relative distance between chin width and mouth width
  - chin width : abs(chin[4] - chin[-5])

1. mouth width: ['top_lip'][0], ['bottom_lip'][0]
2. top-bottom lip distance: ['top_lip'][3], ['bottom_lip'][9]
3,4 angle btw mouth corner to top-bottom lip distance mean

5. left eye size: get max min y
6. right eye size: get max min y
7. relative distance between eye and eyebrow
    - eye distance: ['left_eye'][3], ['right_eye'][0]
    - eyebrow distnace ['left_eyebrow'][-1], ['right_eyebrow'][0]

"""
```

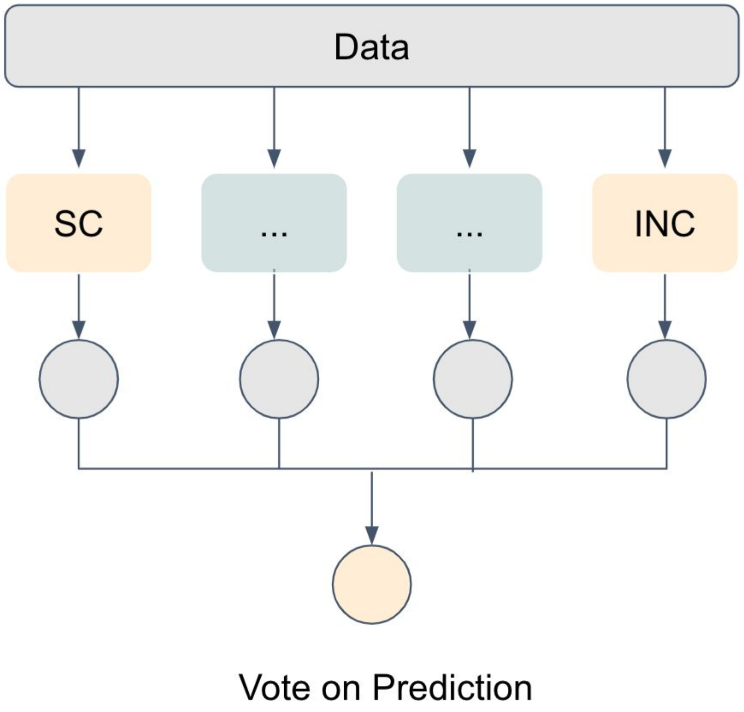

```
('chin', [(10, 25), (10, 29), (12, 33), (13, 36), (15, 40), (17, 43), (19, 46), (22, 48), (26, 48), (30, 48), (34, 45), (38, 42), (40, 38), (42, 33), (42, 28), (41, 23), (40, 18)])
('left_eyebrow', [(9, 21), (10, 19), (11, 19), (13, 19), (15, 20)])
('right_eyebrow', [(20, 18), (23, 17), (25, 16), (28, 15), (31, 16)])
('nose_bridge', [(18, 23), (18, 26), (18, 29), (18, 32)])
('nose_tip', [(17, 34), (19, 35), (20, 35), (22, 34), (24, 33)])
('left_eye', [(12, 24), (13, 22), (15, 22), (17, 24), (15, 24), (13, 25)])
('right_eye', [(24, 22), (25, 20), (27, 20), (30, 20), (28, 22), (26, 22)])
('top_lip', [(18, 40), (19, 39), (20, 38), (22, 39), (24, 38), (27, 38), (30, 38), (29, 38), (24, 39), (22, 40), (20, 40), (19, 40)])
('bottom_lip', [(30, 38), (28, 41), (25, 43), (23, 43), (21, 43), (20, 42), (18, 40), (19, 40), (21, 41), (22, 41), (24, 41), (29, 38)])
```

Mini-VGG

Add an empty image with drawn face landmarks - performance is not good

Ensemble

Mixing models



Soft Vote

Combine the weak classifiers by taking the average of the predicted probability for each class from each classifier.

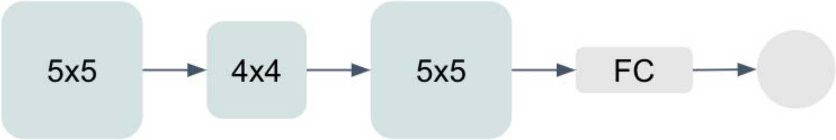
Simple CNN

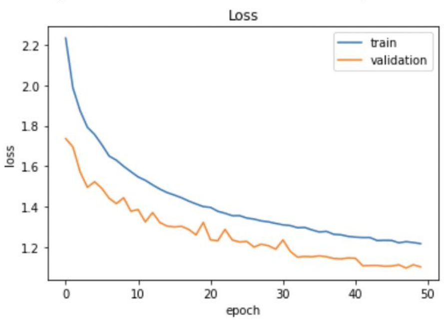
Ensemble

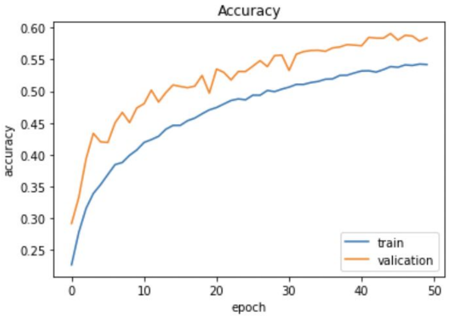
NN = 5 layers with 3 convolutional layers

Accuracy on Testset = 0.54

Total trainable **params** = 69,223








```
NUM_CLASS = 7
lr = 0.0001
opt = Adam(learning_rate=lr)
loss = 'categorical_crossentropy'
metrics = ['accuracy']
n_epochs = 50
```

```
def fiveLayerCNN(input_shape=(IMG_H,IMG_W,IMG_C),
                 num_class=NUM_CLASS):
  num_classes = num_class
  input_shape = input_shape
  model_name = 'fiveLayerCNN'

  inputs = keras.Input(shape = input_shape)
  x = Conv2D(32, 5, activation='elu')(inputs)
  x = BatchNormalization()(x)
  x = MaxPooling2D(3, strides=2)(x)

  x = Conv2D(32, 4, activation='elu')(x)
  x = BatchNormalization()(x)
  x = MaxPooling2D(3, strides=2)(x)

  x = Conv2D(64, 5, activation='elu')(x)
  x = BatchNormalization()(x)
  x = MaxPooling2D(3, strides=2)(x)

  x = Flatten()(x)
  x = Dense(1024, activation='elu')(x)
  x = Dropout(0.3)(x)
  outputs = Dense(num_classes, activation='softmax')(
```

InceptionV3

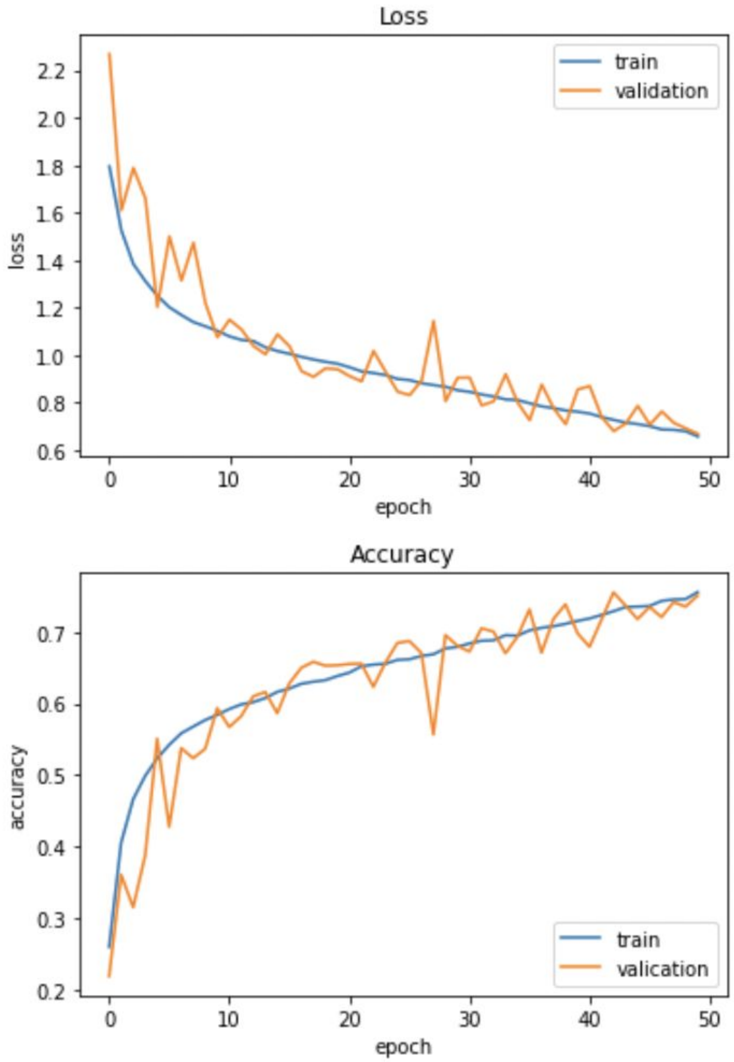
Ensemble

NN = pre-trained on ‘imagenet’

Total trainable **params** = 21,786,797 (x315)

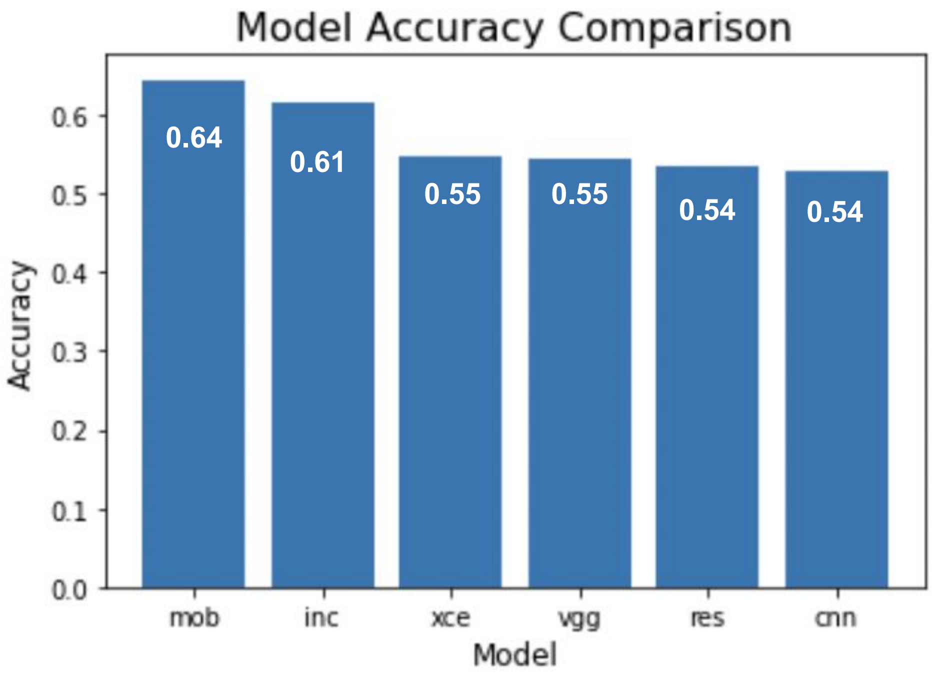
Accuracy on Testset = 0.61 (**+0.9**)

org_img (InputLayer)	[(None, 48, 48, 1)]	0
img_augment (Sequential)	(None, 224, 224, 3)	6
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
batch_normalization_1420 (BatchNormalization)	(None, 5, 5, 2048)	8192
max_pooling2d_163 (MaxPooling2D)	(None, 2, 2, 2048)	0
global_average_pooling2d_41 (GlobalAveragePooling2D)	(None, 2048)	0
dropout_35 (Dropout)	(None, 2048)	0
dense_46 (Dense)	(None, 7)	14343

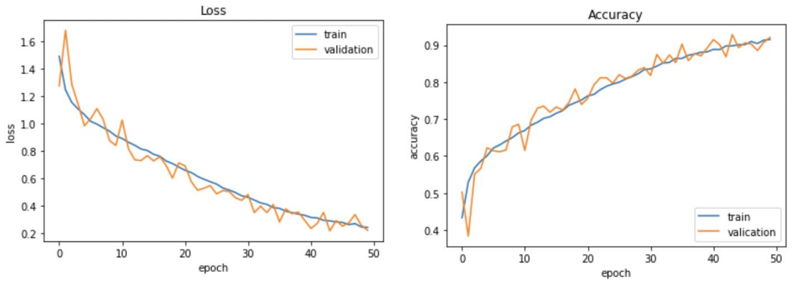


Model Comparisons

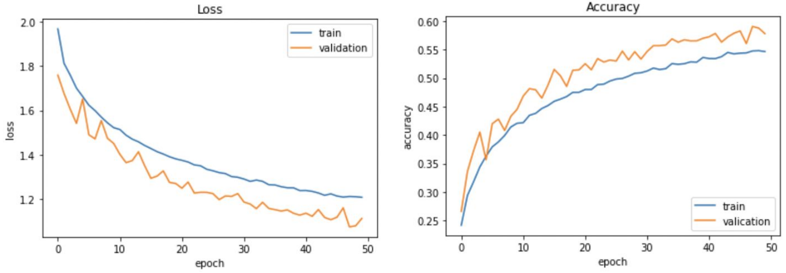
Ensemble



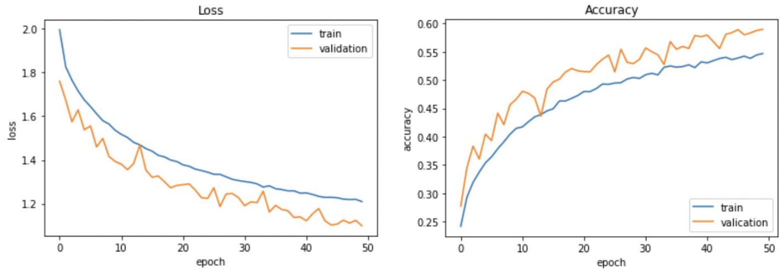
MobileNet



ResNet

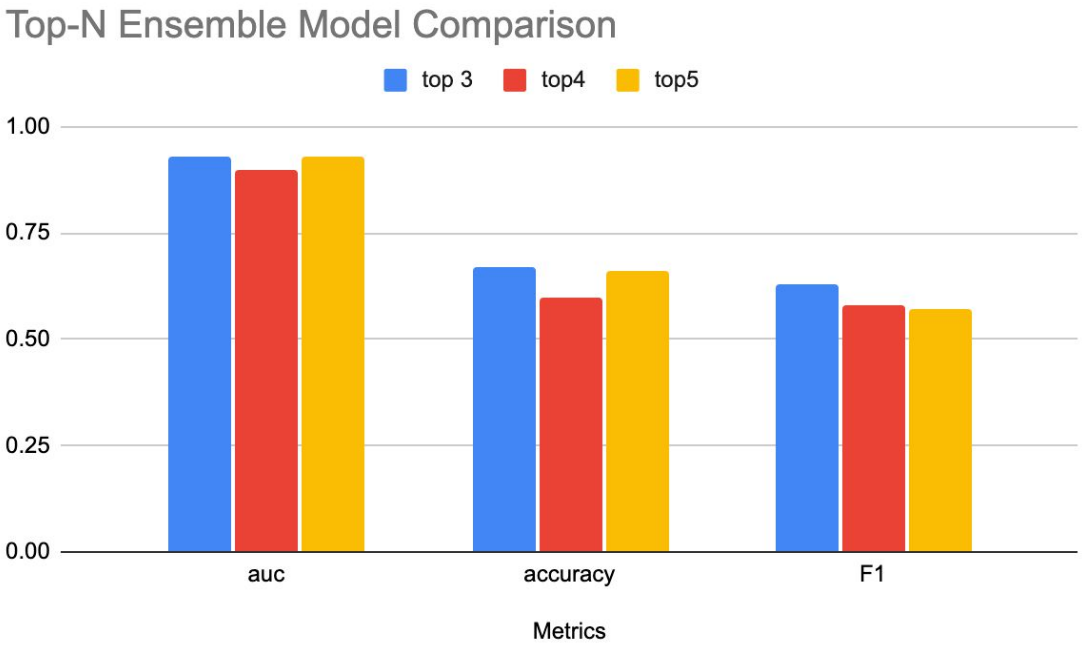


Xception



Top-N Ensembling

Ensemble




```
# top models
simpleCnn = load_model(path +'models/simpleCnn.h5'

mob = load_model(path +'models/Pre_MobileNetV2.h5'
inc = load_model(path +'models/features_InceptionV
xce = load_model(path +'models/Pre_Xception.h5')
vgg = load_model(path +'models/Pre_VGG19.h5')
res = load_model(path +'models/ResNet101V2.h5')
```

```
INP_SIZE = (48,48)
NUM_CLASS = 7

lr = 0.0001
opt = Adam(learning_rate=lr)
act = 'elu'
loss = 'categorical_crossentropy'
metrics = ['accuracy']

def top3(input_shape=(INP_SIZE[0],INP_SIZE[1],1),
    n_classes = n_class
    model_name = 'top3'

    inputs = keras.Input(shape=input_shape)
    y1 = mob(inputs)
    y2 = inc(inputs)
    y3 = xce(inputs)

    outputs = layers.average([y1, y2, y3])

    model = keras.Model(inputs=inputs, outputs=out
    model.compile(loss=loss, optimizer = opt, metr
```

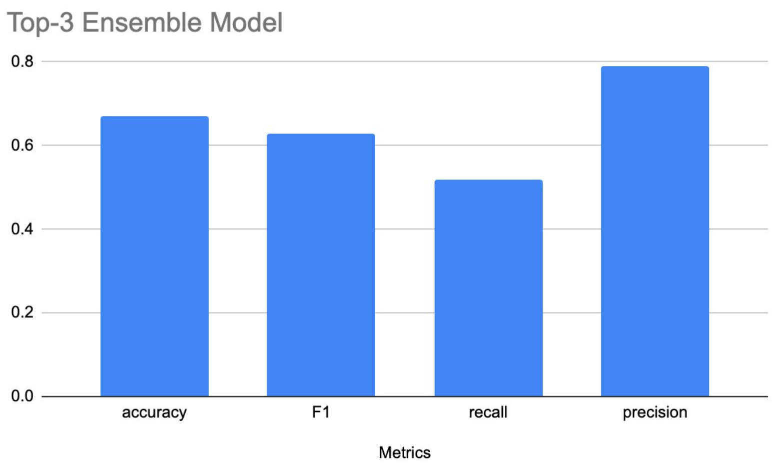
Top-3 Ensembling

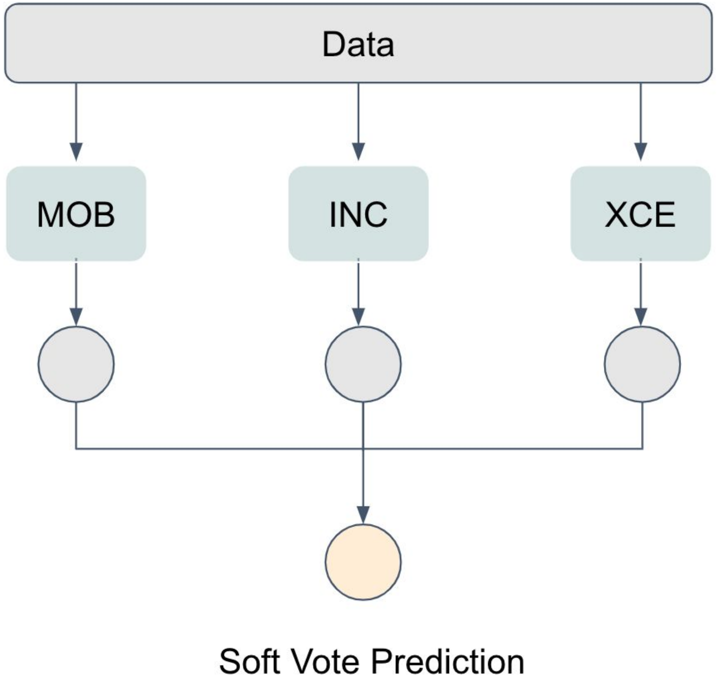
Ensemble Result

NN = ensemble MobileNet, Inception, and Xception

Total trainable **params** = 25,100,129

Accuracy on Testset = 0.67 (+0.13)





Conclusion

- Issue:
  Bad data quality
- Improvement:
  Fine tune, add more epochs, test real images
- Future work:
  Mobile App

Reference

1. A. Ng, “With facial recognition, shoplifting may get you banned in places you’ve never been,” CNET, 20-Mar-2019. [Online]. Available: https://www.cnet.com/tech/services-and-software/with-facial-recognition-shoplifting-may-get-you-banned-in-places-youve-never-been/. [Accessed: 04-Dec-2021]
2. H. K. Sharma *et al.*, “CNN based facial expression recognition system using deep learning approach,” in *Cyber Intelligence and Information Retrieval*, Singapore: Springer Singapore, 2022, pp. 391–405.
3. A. Khanzada, C. Bai, and F. T. Celepcikay, “Facial expression recognition with deep learning,” *arXiv [cs.CV]*, 2020.
4. C. Pramerdorfer and M. Kampel, “Facial expression recognition using Convolutional Neural Networks: State of the art,” *arXiv [cs.CV]*, 2016.
5. J. Tang, X. Zhou, and J. Zheng, “Design of Intelligent classroom facial recognition based on Deep Learning,” J. Phys. Conf. Ser., vol. 1168, p. 022043, 2019

Contribution

Team 2: Mavis Wang, Xiaocen Xie, Coco Yu, Yan Tang, Zuojun Zheng

Facial Expression Recognition Classification on FER2013

Tasks	Intro	Data Collection	Data Preparation	Modeling	ResNet50-V2	mini VGG	Ensembling	Conclusion
Presentation	Xiaocen Xie	Xiaocen Xie	Zuojun Zheng	-->	Zuojun Zheng	Yan Tang	Mavis Wang	Coco Yu
Project	Xiaocen Xie	Zuojun Zheng	Mavis Wang, Zuojun Zheng	-->	Zuojun Zheng	Yan Tang	Mavis Wang	Coco Yu
IEEE Paper	Xiaocen Xie	Yan Tang	Mavis Wang, Zuojun Zheng	All	Zuojun Zheng	Yan Tang	Mavis Wang	Coco Yu