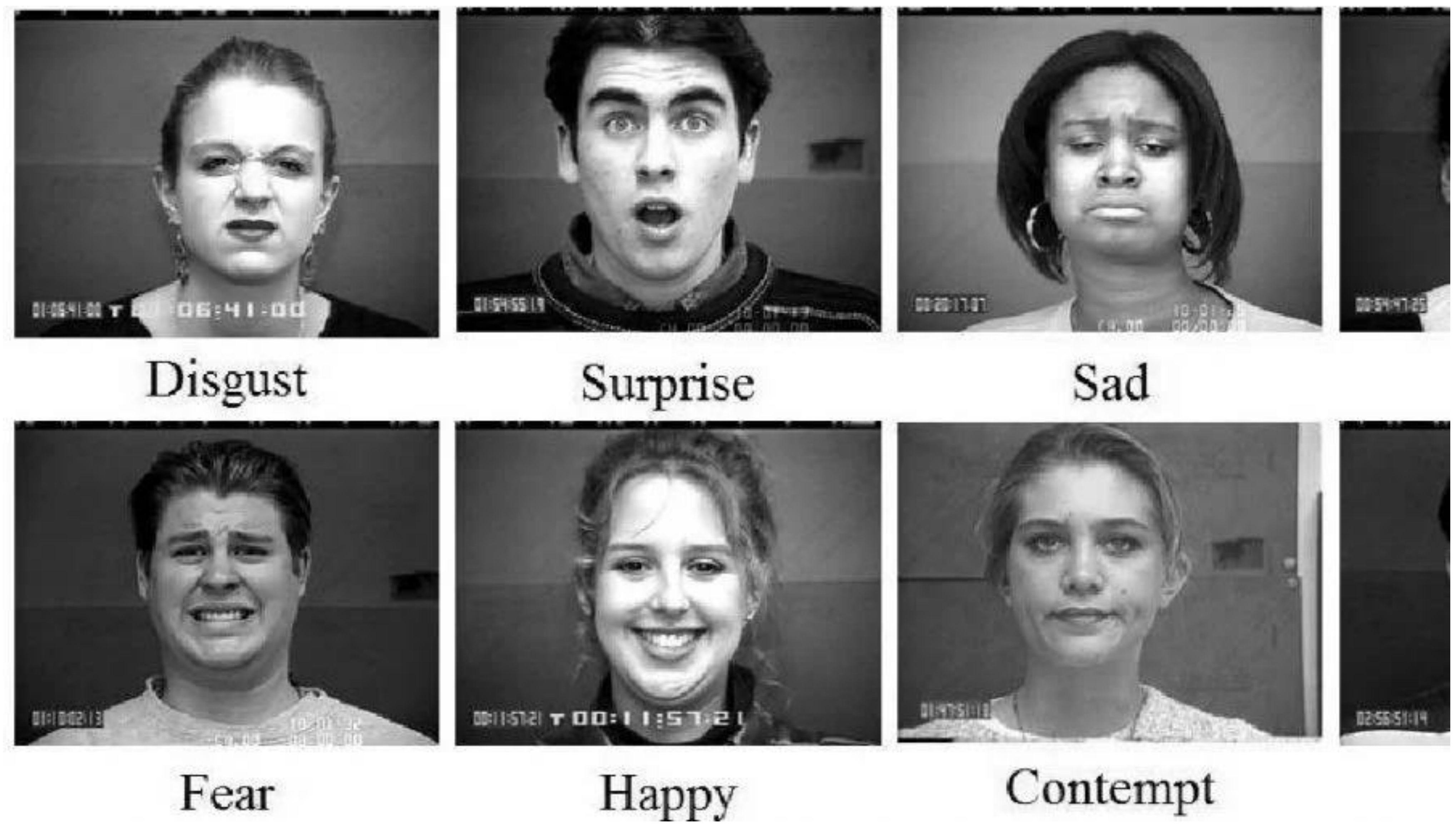
Learning Relation-Aware Facial Expression Representations with Transformers: A Performance Comparison with CNNs

A20521362

Young Sun Lee

What is Emotion Recognition?



What is Emotion Recognition?

- ***Definition:*** A technology that analyzes facial expressions to identify human emotions such as happiness, sadness, surprise, etc.
- ***Importance:*** This can contribute to **human-computer interaction** and the empathic abilities of future robots.
- ***Applications:*** Digital advertising, online gaming, customer feedback analysis, healthcare.
- ***Examples:***

  UAE: Assessing the emotional atmosphere of the population in public spaces.

  Skyscanner: Measuring and improving customer satisfaction.
- ***Challenges:***

  The rigorous definition of the relationship between expressions and emotions.

  Ethical issues such as bias promotion, as well as privacy concerns.

  The problem of fraud due to misuse.

Introduction to the FER-2013 Dataset

Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.

Introduction to the FER-2013 Dataset

- ***Release:*** The FER-2013 dataset is an open-source dataset originally created by Pierre-Luc Carrier and Aaron Courville for their ongoing project and was publicly shared in preparation for a competition on Kaggle just before ICML-2013.
- ***Classes:*** The dataset is categorized into seven emotions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.
- ***Image Size:*** The images are 48x48 pixels.
- ***Color:*** black and white
- ***Number of Data:*** There are 28,709 training images and 3,589 test images.
- ***Features:*** The dataset includes off-frontal poses and various lighting conditions, making the pictures relatively natural. The number of images is not the same across different emotions.

- Human performance on this dataset is known to be around **65.5%.**

Data Transformation, Augmentation, and Data Loader Configuration

- ***Resizing:*** Resize images to 230x230 dimensions.
- ***Data Augmentation:***

  Random Rotation (±15°)

  Random Crop (224x224 with padding)

  Random Horizontal Flip
- ***Tensor Transformation:*** Convert images to PyTorch tensors.
- ***Batch Size:*** 32
- ***Shuffling:*** Training data is shuffled, test data is not shuffled.

Models compared in this project

- AlexNet
- VGGNet
- ResNet-50
- TransFER
- DACL

AlexNet

- 1st place in 2012 ILSVRC (performance difference of more than 10% over the second place)
- Five convolutional layers followed by three fully-connected layers
- Activation function: ReLU
- overlapping pooling, parallel use of two GPUs, local response normalization

AlexNet – this project

- 3 channel color photo of 256x256x3 → 224x224x1 (FER-2013 consists of grayscale images)

Model	**Data Augmentation**	**Optimizer**	**Learning Rate**	**Epoch**	**Accuracy**	**Note**
AlexNet	Yes	Adam	0.001	40	**49.77%**	
AlexNet	Yes	SGD, Momentum 0.9	0.001	40	**40.71%**	
AlexNet	No	Adam	0.001	40	**25.13%**	No Improvement

VGGNet

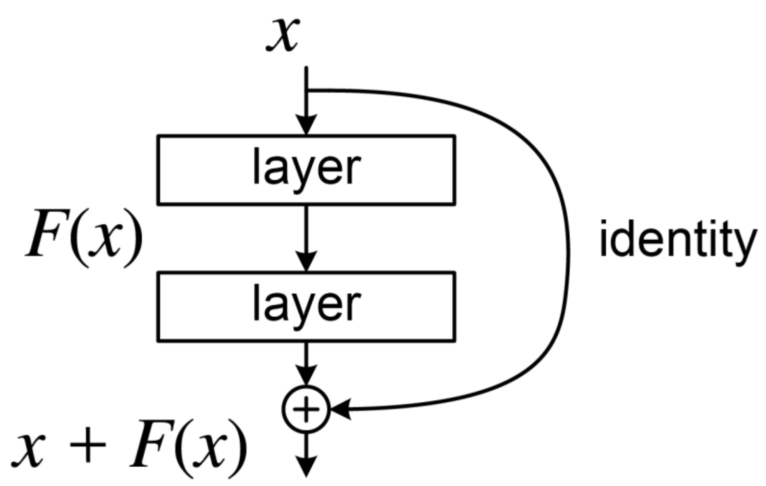
- Second in the 2014 ILSVRC (GoogleNet won the first place)
- Increased depth to enhance performance. (simple structure)
- 'Very Deep Convolutional Networks For Large-Scale Image Recognition’ (compared six different architectures)
- Used only 3x3 size filters.
- The basic architecture: Conv layer - max-pooling to reduce resolution
- Started with 64 channels and doubled them each time the resolution was reduced by max-pooling.
- Three FC (Fully Connected) layers followed by a softmax layer.

VGGNet – this project

- Adjusted version of VGGNet-16 to fit the current input
- Composed of four convolutional stages
- Each stage consists of two convolutional layers followed by max pooling
- After each convolution, batch normalization and ReLU activation are used
- The convolutional layers are followed by three fully connected layers before reaching the final classification layer

Model	Data Augmentation	Optimizer	Learning Rate	Epoch	Accuracy	Note
VGGNet	Yes	SGD, Momentum 0.9	0.001	40	**66.19%**	
VGGNet	Yes	Adam	0.001	40	**66.10%**	

ResNet-50 (Residual neural network)



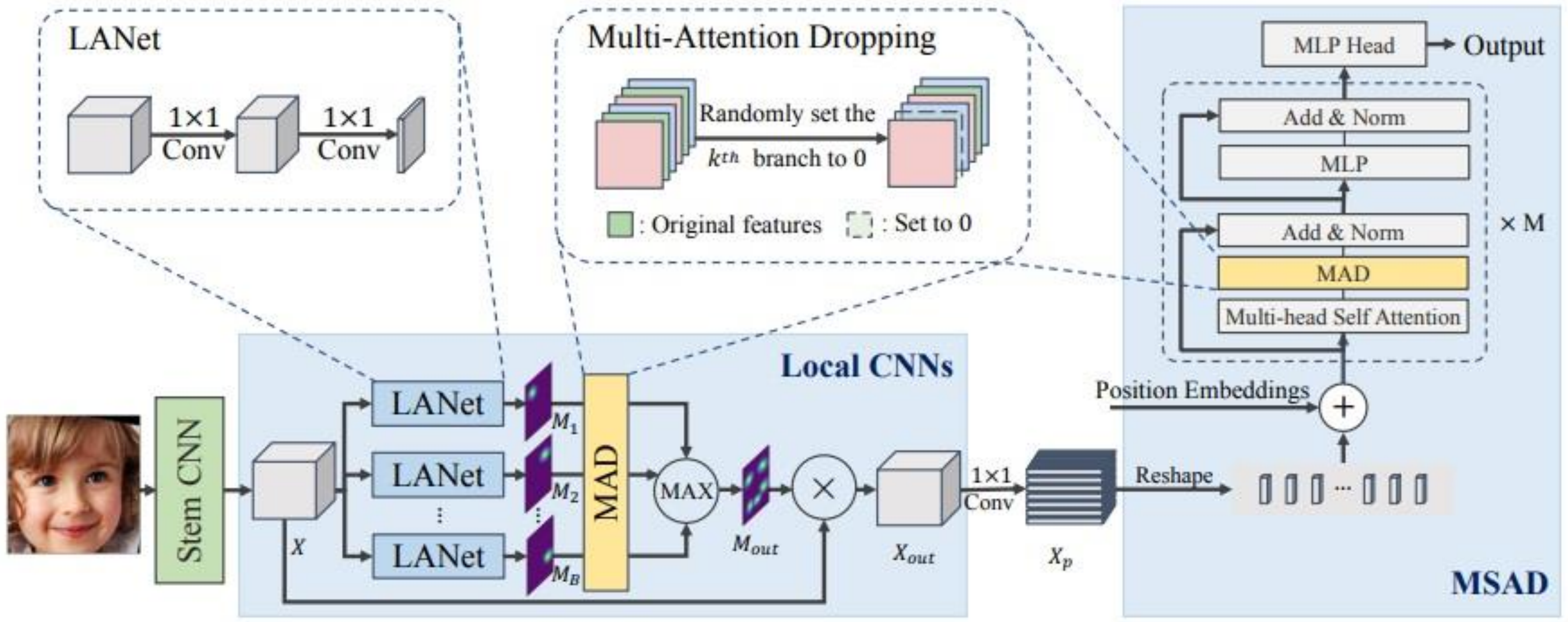
- Won the 2015 ILSVRC
- Top-5 error rate in the competition was 3.57% (first neural network that surpassed humans(5%))
- Very deep neural network (mitigate the problems of vanishing gradient or exploding gradient)
- ResNet allowed connections between the i-th and (i+r)-th layers (shortcut connection)
- By using shortcut connection, gradient can be easily propagated to the previous layer during backpropagation

ResNet-50 – this project

- ResNet comes in various versions like 18, 34, 50, 101, and 152, and for this project, I utilized **ResNet-50**, whi ch was modified to accept an input of 224x224x1.

Model	**Data Augmentation**	**Optimizer**	**Learning Rate**	**Epoch**	**Accuracy**	**Note**
ResNet-50	Yes	Adam	0.001	40	**66.73%**	
ResNet-50	Yes	SGD, Momentum 0.9	0.001	40	**57.95%**	

TransFER


LANet
1×1 Conv
1×1 Conv
Multi-Attention Dropping
Randomly set the k^{th} branch to 0
: Original features
: Set to 0
MLP Head
Output
Add & Norm
MLP
Add & Norm
MAD
Multi-head Self Attention
× M
Position Embeddings
Stem CNN
Local CNNs
LANet
LANet
LANet
M_1
M_2
M_B
MAD
MAX
M_{out}
X
X_{out}
1×1 Conv
X_p
Reshape
MSAD

TransFER

1. Facial images are first fed into a ***stem CNN*** to extract feature maps (IR-50)

2. Feature maps are then passed through the ***local CNNs*** to locate diverse useful feature areas.

3. ***1 × 1 convolution and reshape operations*** are used to project feature maps to a sequence of feature vectors which can be directly input into MSAD (MAD in a Transformer encoder)

4. ***MSAD* (MAD in a Transformer encoder)** pushes multi-head self-attention to explore rich relations among different local patches.

5. Optimizer: ***SGD***

- ***MAD(Multi-Attention Dropping):*** proposed to prevent neural networks from overfitting. In contrast with the standard Dropout, MAD adopts a group of feature maps (or vectors) as input and treats every feature map as a whole. This means that one feature map is selected from a uniform distribution and is entirely set to zeroes.

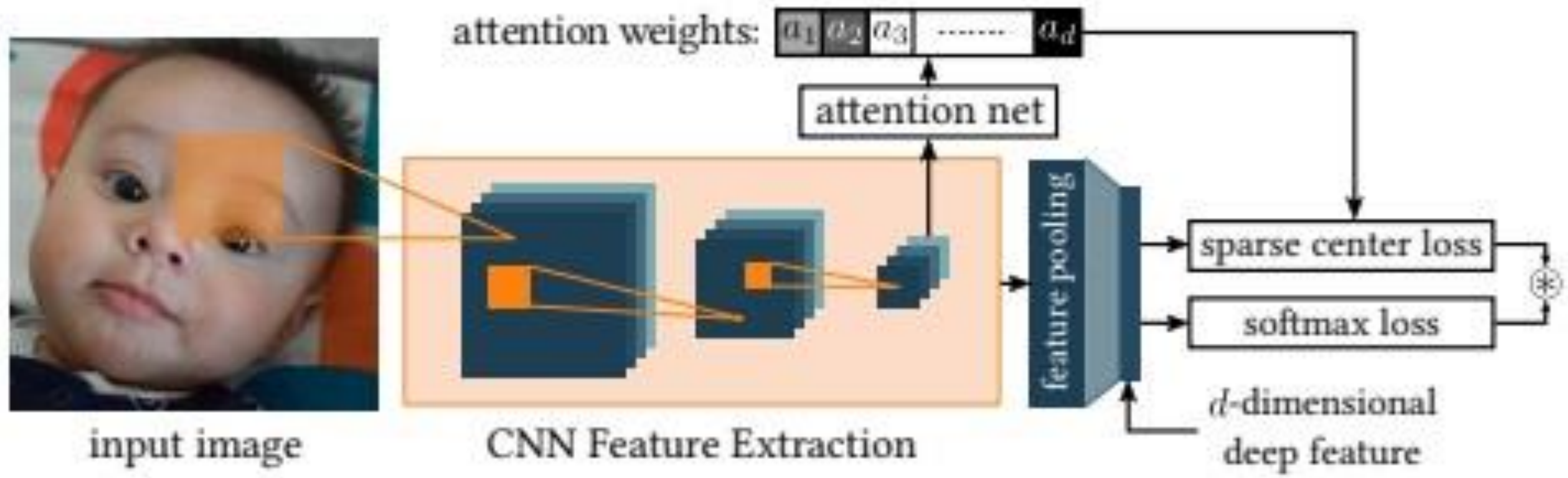
TransFER – this project
(ResNet-18 with Transformer and Dropouts)

- ***Stem CNN:*** ResNet-18 (ResNet-50 exceeded memory capacity of the Colab A100)
- ***Local CNNs :*** CNN1, CNN2
- ***MAD: Standard dropout***
- ***Transformer:***
  The encoder is transformer which doesn't have a MAD layer

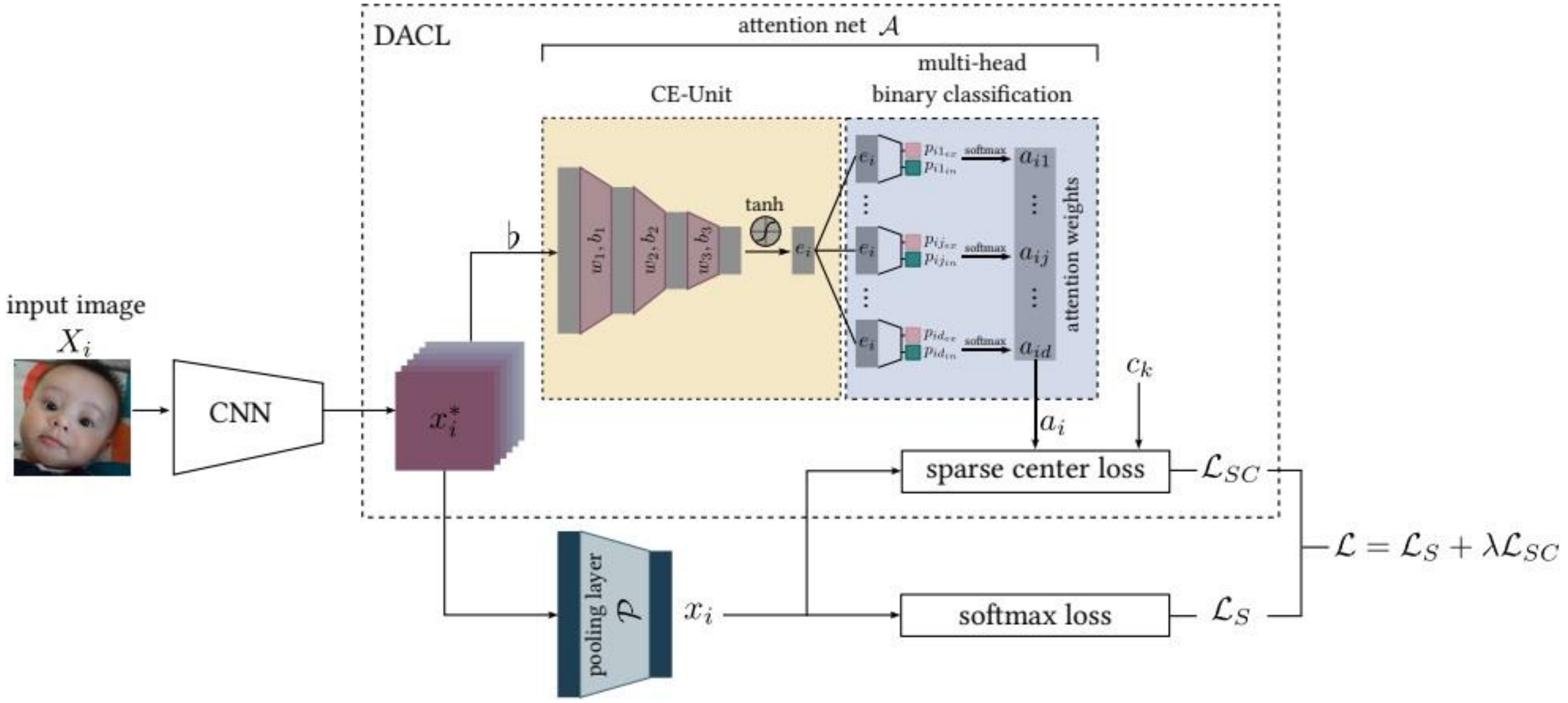
```
def forward(self, x):
    x = self.stem_cnn(x)
    x1 = self.local_cnn1(x)
    x2 = self.local_cnn2(x)
    x1 = self.mad(x1)
    x2 = self.mad(x2)
    x = torch.cat((x1, x2), dim=2)
    x = x.permute(0, 2, 3, 1).flatten(1, 2)
    x = self.transformer_encoder(x)
    x = self.classifier(x.mean(dim=1))
    return x
```

Model	Data Augmentation	Optimizer	Learning Rate	Epoch	Accuracy	Notes
TransFER	Yes	SGD, Momentum 0.9	0.001	40	**67.58%**	
TransFER	Yes	SGD, Momentum 0.9	0.001	30	**67.39%**	
TransFER	Yes	SGD, Momentum 0.9	0.001	10	**64.43%**	
TransFER	Yes	Adam	0.0001	40	**25.13%**	No Improvement
TransFER	Yes	Adam	0.001	40	**25.07%**	No Improvement

DACL


attention weights:
a_1 a_2 a_3 .... a_d
attention net
feature pooling
sparse center loss
softmax loss
$\circledast$
d-dimensional deep feature
input image
CNN Feature Extraction

DACL


DACL
attention net $\mathcal{A}$
multi-head binary classification
CE-Unit
input image X_i
CNN
x_i^*
w_1, b_1
w_2, b_2
w_3, b_3
tanh
e_i
$p_{i1_{ex}}$
$p_{i1_{in}}$
$p_{ij_{ex}}$
$p_{ij_{in}}$
$p_{id_{ex}}$
$p_{id_{in}}$
softmax
a_{i1}
a_{ij}
a_{id}
attention weights
a_i
c_k
sparse center loss
$\mathcal{L}_{SC}$
$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_{SC}$
pooling layer $\mathcal{P}$
x_i
softmax loss
$\mathcal{L}_S$

DACL

- The CNN is Pre-trained ResNet18. This yiels the convolutional spatial feature map.
- The CE-Unit takes the spatial feature map as a context and yields **an encoded latent feature vector** to eliminate noise and irrelevant information
- The multi-head binary classification module then calculates the **attention weight**.
- The final loss is a combination of ***softmax loss*** $\boldsymbol{L_s}$ and ***sparse center loss*** $\boldsymbol{L_{sc}}$
- Optimizer: ***SGD***

DACL – this project
(ResNet-18 with Attention)

1. The input is transformed into significant features in the base model (ResNet-18)
2. The dimensions of the feature map are altered before passing it to the attention module
3. The attention module enables focusing on the most important features, enhancing the model's predictive capability
4. The tensor is flattened and then passed through a fully connected layer
5. the part where the loss is combined, as in the paper, was not implemented.

```
def forward(self, x):
    x = self.base_model(x)
    x = x.view(x.size(0), 512, 1, 1)
    x = self.attention(x)
    x = x.view(x.size(0), -1)
    x = self.fc(x)
    return x
```

Model	Data Augmentation	Optimizer	Learning Rate	Epoch	Accuracy	Note
DACL	Yes	Adam	0.001	40	**69.08%**	
DACL	Yes	SGD, Momentum 0.9	0.001	40	**67.96%**	

Accuracy Ranking

No.	Model	Data Augmentation	Optimizer	Learning Rate	Epoch	Accuracy	Note
1	**DACL**	Yes	Adam	0.001	40	**69.08%**	
2	**DACL**	Yes	SGD, Momentum 0.9	0.001	40	**67.96%**	
3	**TransFER**	Yes	SGD, Momentum 0.9	0.001	40	**67.58%**	
4	**TransFER**	Yes	SGD, Momentum 0.9	0.001	30	**67.39%**	
5	**ResNet-50**	Yes	Adam	0.001	40	**66.73%**	
6	**VGGNet**	Yes	SGD, Momentum 0.9	0.001	40	**66.19%**	
7	**VGGNet**	Yes	Adam	0.001	40	**66.10%**	
8	**TransFER**	Yes	SGD, Momentum 0.9	0.001	10	**64.43%**	
9	**ResNet-50**	Yes	SGD, Momentum 0.9	0.001	40	**57.95%**	
10	**AlexNet**	Yes	Adam	0.001	40	**49.77%**	
11	**AlexNet**	Yes	SGD, Momentum 0.9	0.001	40	**40.71%**	
12	**CunstomCNN +Transformer**	Yes	Adam	0.0001	40	**25.52%**	No Improvement
13	**TransFER**	Yes	Adam	0.0001	40	**25.15%**	No Improvement
14	**AlexNet**	No	Adam	0.001	40	**25.13%**	No Improvement
15	**CunstomCNN +Transformer**	Yes	Adam	0.001	40	**25.12%**	No Improvement
16	**TransFER**	Yes	Adam	0.001	40	**25.07%**	No Improvement
17	**CunstomCNN +Transformer**	Yes	SGD, Momentum 0.9	0.001	10	**24.71%**	No Improvement