DR. D. Y PATIL INSTITUTE OF ENGINEERING MANAGEMENT AND RESEARCH

Department Of Artificial Intelligence and Data Science

AY 2022-23 SEMESTER 5

THIRD YEAR ENGINEERING

Knowledge Distillation and Student Teacher Learning for Visual Intelligence

Guide

Mrs. Pallavi Yevale

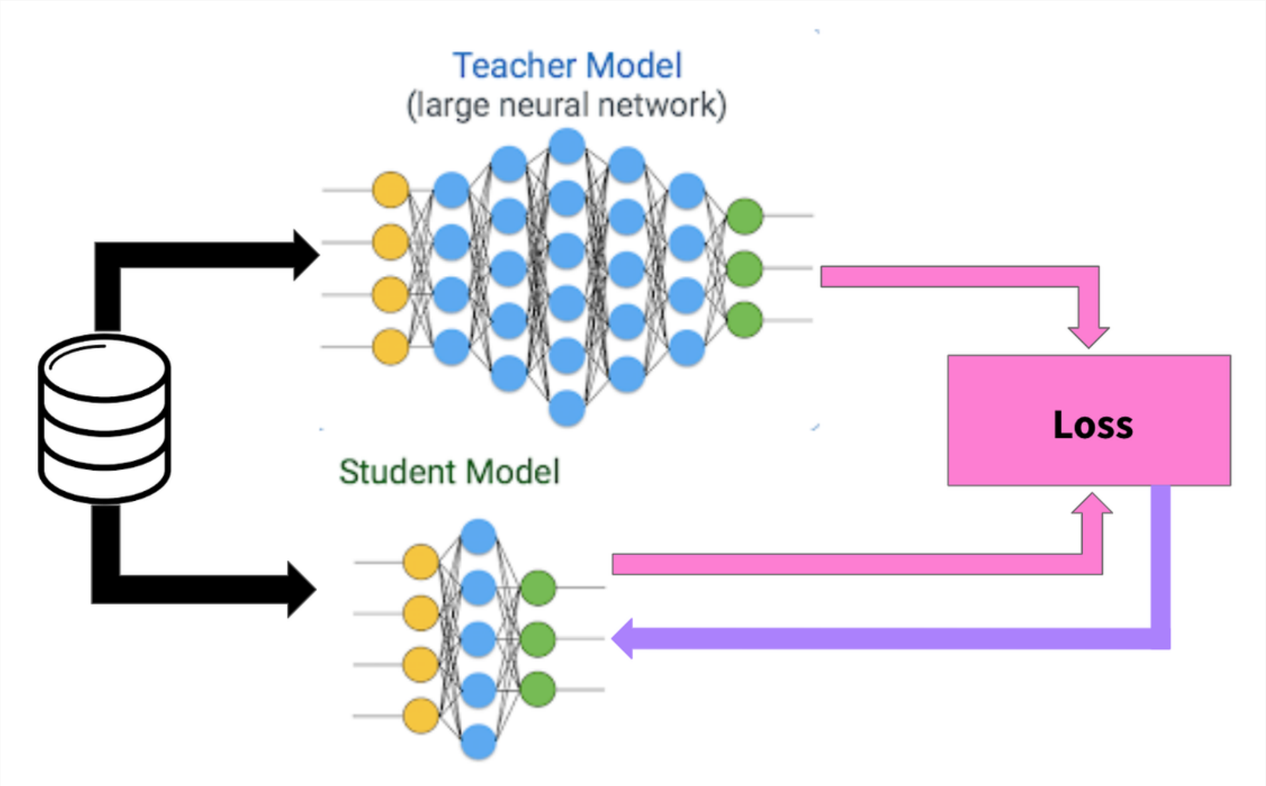
Student: Saniya Mulla

Roll no: 30

Table of Contents

I ABSTRACT

- Deep neural models solve very complex problems.
- However, these models are huge in size with millions of parameters, demanding heavy computation power and failing to be deployed on edge devices.
- Besides, the performance boost is highly dependent on redundant labeled data.
- To achieve faster speeds and to handle problems caused by the lack of labeled data, knowledge distillation (KD) has been proposed to transfer information learned from one model to another.
- KD is often characterized by the so-called 'Student-Teacher' (S-T) learning framework and has been broadly applied in model compression and knowledge transfer.



II INTRODUCTION

- Refers to the idea of model compression by teaching a smaller network, step by step, exactly what to do using a bigger already trained network.
- The ‘soft labels’ refer to the output feature maps by the bigger network after every convolution layer.
- The smaller network is then trained to learn the exact behavior of the bigger network by trying to replicate it’s outputs at every level (not just the final loss).

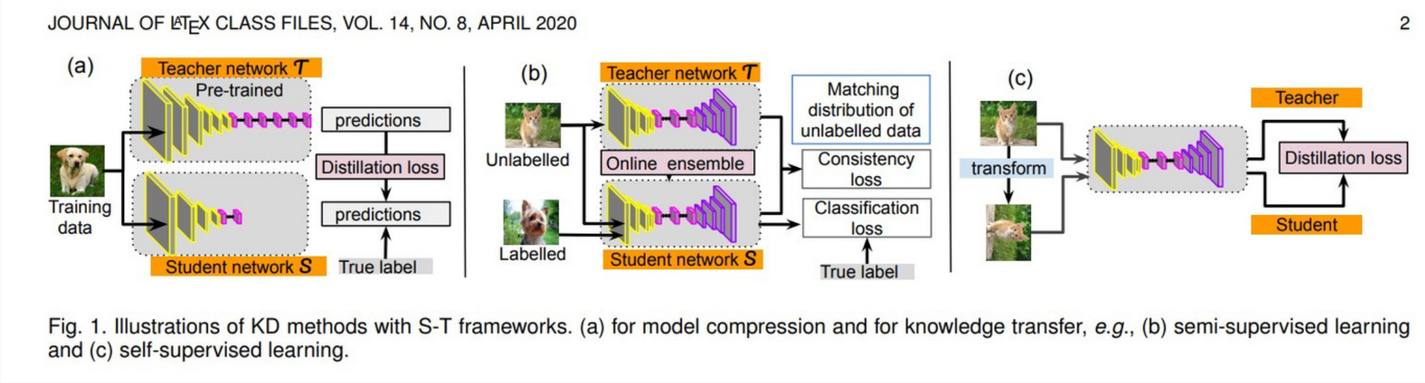
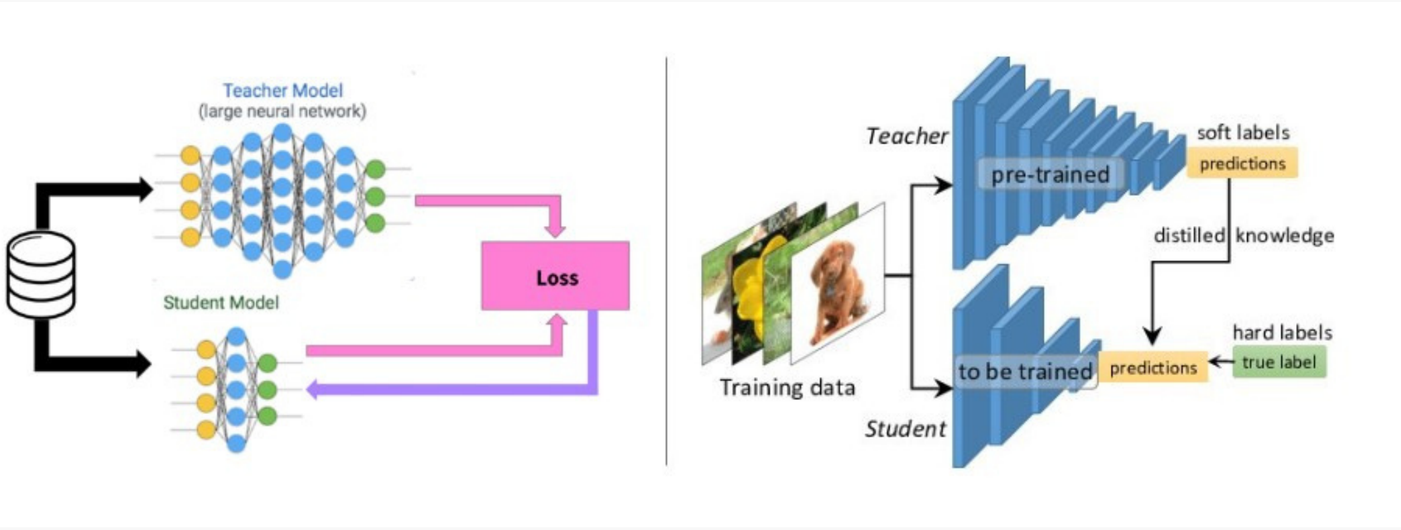
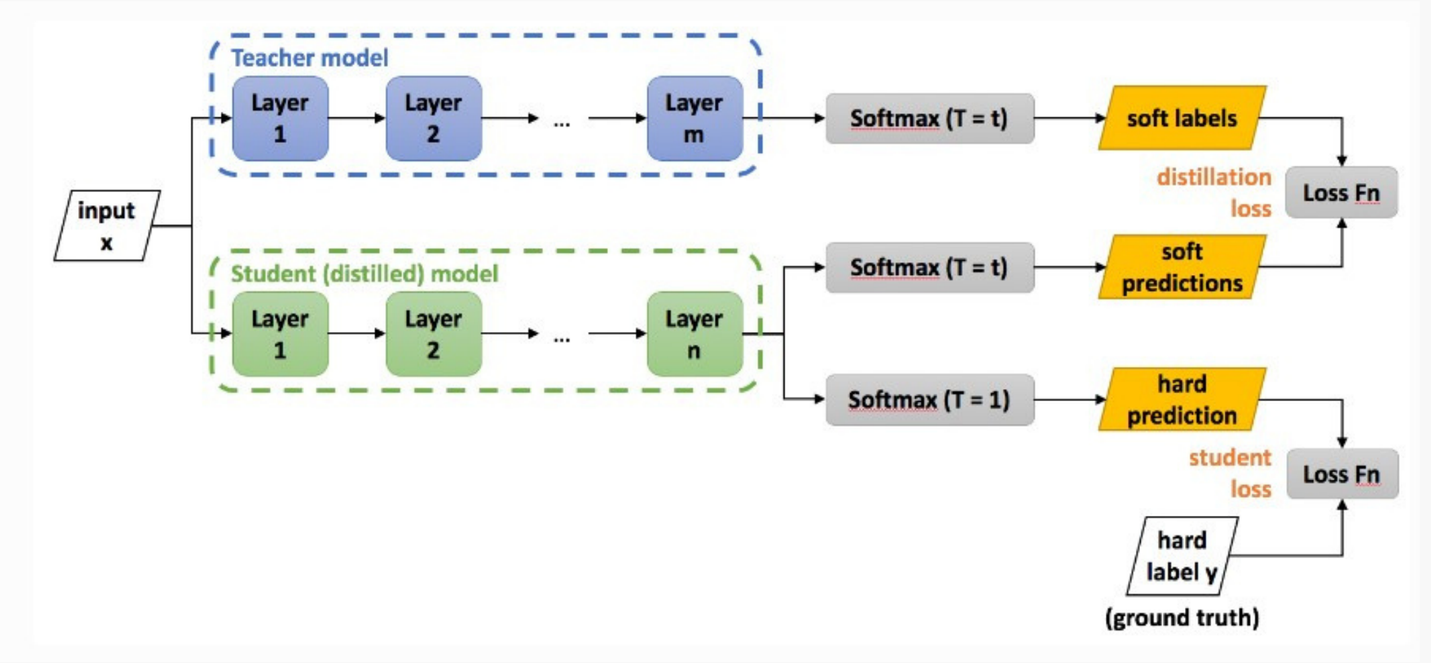


Fig. 1. Illustrations of KD methods with S-T frameworks. (a) for model compression and for knowledge transfer, *e.g.*, (b) semi-supervised learning and (c) self-supervised learning.



III METHODOLOGY

- Knowledge is transferred by minimizing a loss function in which the target is the distribution of class probabilities predicted by the teacher model.
- That is - the output of a softmax function on the teacher model's logits.
- As such, it doesn't provide much information beyond the ground truth labels already provided in the dataset.
- Hinton et al., 2015 introduced the concept of "***softmax temperature***".

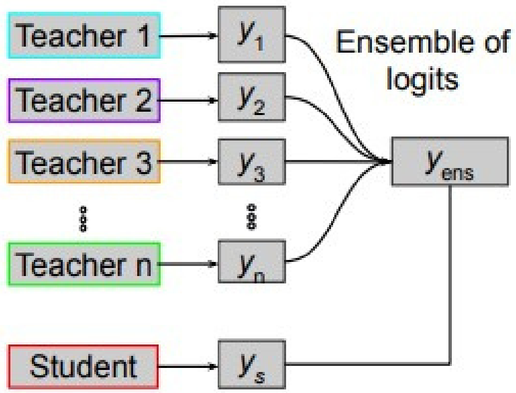


The probability pi of class i is calculated from the logits z as:

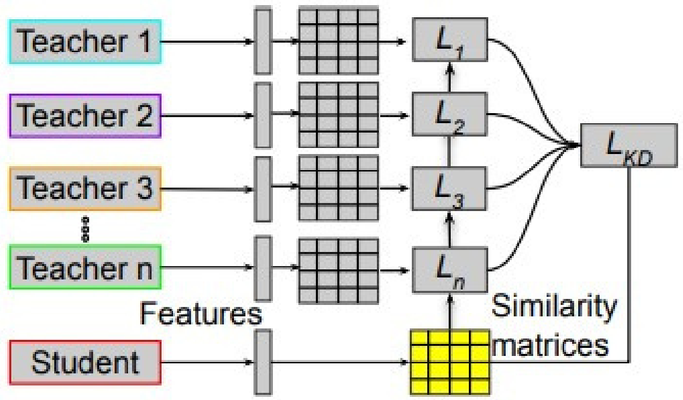
$$p_i = \frac{exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

where T is the Temperatire parameter.

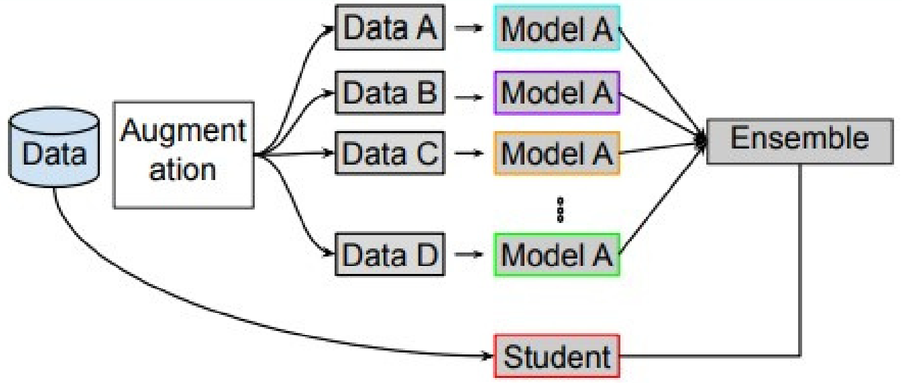
DISTILLATION FROM MULTIPLE TEACHERS



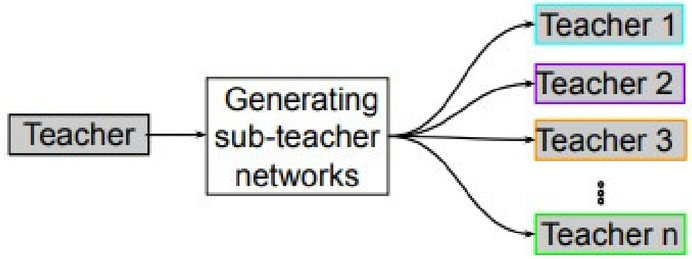
(a) ensemble of logits



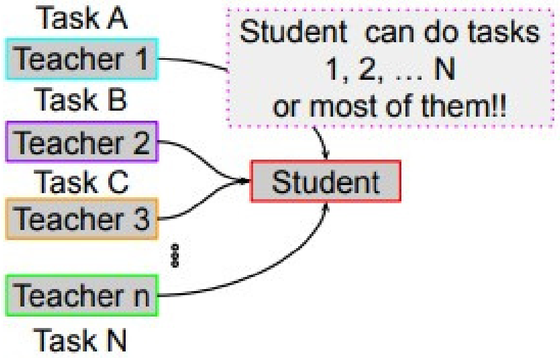
(b) ensemble of features



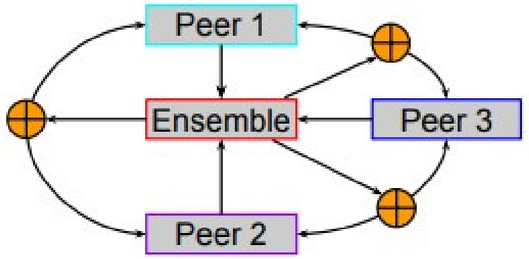
(c) unifying data sources



(e) obtaining sub-teacher networks from singe teacher



(f) customizing student from heterogeneous teachers



(g) Mutual learning with ensemble of peers

IV IMPLEMENTATION OF THE PROPOSED MODEL:

TEACHER MODEL

```
[3]  teacher = tf.keras.Sequential(
        [
            tf.keras.Input(shape=(32, 32, 3)),

            tf.keras.layers.Conv2D(256, (3, 3), strides=(2, 2), padding="same"),
            tf.keras.layers.LeakyReLU(alpha=0.2),
            tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
            tf.keras.layers.Conv2D(512, (3, 3), strides=(2, 2), padding="same"),
            tf.keras.layers.LeakyReLU(alpha=0.2),
            tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(10),
        ],
        name="teacher",
    )
    teacher.summary()
```

```
Model: "teacher"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 16, 16, 256)       7168

 leaky_re_lu (LeakyReLU)     (None, 16, 16, 256)       0

 max_pooling2d (MaxPooling2D  (None, 16, 16, 256)      0
 )

 conv2d_1 (Conv2D)           (None, 8, 8, 512)         1180160

 leaky_re_lu_1 (LeakyReLU)   (None, 8, 8, 512)         0

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 512)        0
 2D)

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 10)                327690

=================================================================
Total params: 1,515,018
Trainable params: 1,515,018
Non-trainable params: 0
_________________________________________________________________
```

STUDENT MODEL

```
[4]  # Create the student
     student = tf.keras.Sequential(
         [
             tf.keras.Input(shape=(32, 32, 3)),
             tf.keras.layers.Conv2D(64, (3, 3), strides=(2, 2), padding="same"),
             tf.keras.layers.LeakyReLU(alpha=0.2),
             tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
             tf.keras.layers.Conv2D(256, (3, 3), strides=(2, 2), padding="same"),
             tf.keras.layers.LeakyReLU(alpha=0.2),
             tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding="same"),
             tf.keras.layers.Flatten(),
             tf.keras.layers.Dense(10),
         ],
         name="student",
     )
     student.summary()
```

```
Model: "student"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 16, 16, 64)        1792

 leaky_re_lu_2 (LeakyReLU)   (None, 16, 16, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 8, 8, 256)         147712

 leaky_re_lu_3 (LeakyReLU)   (None, 8, 8, 256)         0

 max_pooling2d_3 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 flatten_1 (Flatten)         (None, 16384)             0

 dense_1 (Dense)             (None, 10)                163850

=================================================================
Total params: 313,354
Trainable params: 313,354
Non-trainable params: 0
_________________________________________________________________
```

TEACHER MODEL:

```
teacher.compile(
    optimizer=tf.keras.optimizers.Adam(),    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],
)
# Train and evaluate teacher on data.
teacher.fit(x_train, y_train, epochs=5)
teacher.evaluate(x_test, y_test)
```

```
Epoch 1/5
1563/1563 [==============================] - 604s 386ms/step - loss: 1.3376 - sparse_categorical_accuracy: 0.5310
Epoch 2/5
1563/1563 [==============================] - 598s 382ms/step - loss: 1.0206 - sparse_categorical_accuracy: 0.6507
Epoch 3/5
1563/1563 [==============================] - 593s 380ms/step - loss: 0.9051 - sparse_categorical_accuracy: 0.6952
Epoch 4/5
1563/1563 [==============================] - 601s 384ms/step - loss: 0.7981 - sparse_categorical_accuracy: 0.7317
Epoch 5/5
1563/1563 [==============================] - 607s 388ms/step - loss: 0.6909 - sparse_categorical_accuracy: 0.7633
313/313 [==============================] - 29s 92ms/step - loss: 1.0497 - sparse_categorical_accuracy: 0.6788
[1.0497170686721802, 0.6787999868392944]
```

We can see that the teacher model takes **604s** per epoch to train and gives an accuracy of **67.88%**

STUDENT MODEL:

```
# Distill teacher to student
distiller.fit(x_train, y_train, epochs=5)
# Evaluate student on test dataset
distiller.evaluate(x_test, y_test)
```

```
 1/5
er prediction   ... Tensor("teacher/dense/BiasAdd:0", shape=(None, 10), dtype=float32)
in distiller : Tensor("add:0", shape=(), dtype=float32)
... {'sparse_categorical_accuracy': <tf.Tensor 'Identity:0' shape=() dtype=float32>, 'student_loss': <tf.Tensor 'sparse_categorical_crossentropy/weighted_loss/value:0' shape=() dtype=fl
er prediction   ... Tensor("teacher/dense/BiasAdd:0", shape=(None, 10), dtype=float32)
in distiller : Tensor("add:0", shape=(), dtype=float32)
... {'sparse_categorical_accuracy': <tf.Tensor 'Identity:0' shape=() dtype=float32>, 'student_loss': <tf.Tensor 'sparse_categorical_crossentropy/weighted_loss/value:0' shape=() dtype=fl
1563 [==============================] - 257s 164ms/step - sparse_categorical_accuracy: 0.5427 - student_loss: 1.3019 - distillation_loss: 0.0842
 2/5
1563 [==============================] - 253s 162ms/step - sparse_categorical_accuracy: 0.6729 - student_loss: 0.9593 - distillation_loss: 0.0449
 3/5
1563 [==============================] - 254s 162ms/step - sparse_categorical_accuracy: 0.7229 - student_loss: 0.8169 - distillation_loss: 0.0348
 4/5
1563 [==============================] - 257s 164ms/step - sparse_categorical_accuracy: 0.7534 - student_loss: 0.7207 - distillation_loss: 0.0307
 5/5
1563 [==============================] - 254s 162ms/step - sparse_categorical_accuracy: 0.7855 - student_loss: 0.6275 - distillation_loss: 0.0285
.. {'sparse_categorical_accuracy': <tf.Tensor 'Identity:0' shape=() dtype=float32>, 'student_loss': <tf.Tensor 'sparse_categorical_crossentropy/weighted_loss/value:0' shape=() dtype=flo
13 [==============================] - 6s 18ms/step - sparse_categorical_accuracy: 0.7267 - student_loss: 0.8614
67000079154968, 0.9821649789810181]
```

Whereas the Student distilled model takes **257s** per epoch to train and gives an accuracy of **72.67%**

V BENEFITS

1. Can be deployed at the Edge:

Knowledge distillation, a student model, inherits better quality from the teacher and is more efficient for inference due to its compactness needing less computational resources.

2. Can be used when there is less training data:

Can be used even when there is lesser training data available for the student model. It is enough to have a well trained teacher.

3. Improves generalization:

Soft targets have less variance allowing student networks to be trained on much less data than the Teacher using a much higher learning rate.

4. Less Computation Power:

Less compute requirements and superior performance under stringent production constraints.

VI APPLICATIONS

1. Semantic and Motion Segmentation

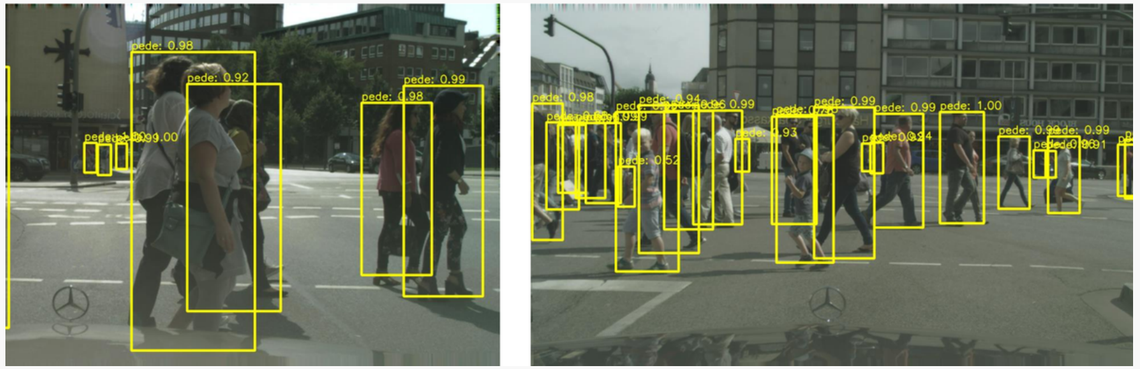
Special classification problem that predicts the category label in pixel wise manner, large model sizes and high computation costs, KD helps to train lightweight networks.

2. Generic Object Detection

Learned a student detector by distilling knowledge from the intermediate layers, logits and regressor of the teacher, in contrast to previous models in which only the intermediate layers of the teacher was utilized to identify informative locations.

3. Pedestrian Detection

various sizes and aspect ratios of pedestrians under extreme illumination conditions are challenges. we use the unified hierarchical knowledge via multiple intermediate supervisions in which the logits information are distilled.



VII Conclusion

- Knowledge Distillation can compress a Deep CNN while maintaining the accuracy so that it can be deployed on embedded systems that have less storage and computational power.
- By distilling Knowledge from a Teacher Network having 27M parameters to a Student Network having only 0.296M parameters (almost 100 times lighter), we were able to achieve almost the same accuracy.
- With more hyperparameter iterations and ensembling of multiple students networks as mentioned in reference [3], the performance of the student model can be further improved.

Discussions

- Are bigger models better teachers?
- Is a pretrained teacher important?
- Single Techer Vs Multiple teachers

VIII REFERENCES

- M.-C. Wu and C.-T. Chiu, “Multi-teacher knowledge distillation for compressed video action recognition based on deep learning,” Journal of Systems Architecture, vol. 103, p. 101695, 2020.

- Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, “Distillating knowledge from graph convolutional networks,” arXiv preprint arXiv:2003.10477, 2020.

- Z.-W. Hong, P. Nagarajan, and G. Maeda, “Periodic intra ensemble knowledge distillation for reinforcement learning,” arXiv preprint arXiv:2002.00149, 2020

Friday, 16/09/2022

AI&DS, DYPIEMR

ADVISOR

Mrs. Pallavi Yevale

STUDENT

Saniya Mulla

Roll no - 30

Thank you for listening!