Project 3: Weakly supervised learning: label noise and correction

Team Group 7

Team members

- Guosheng Cai
- Jun Ding
- Zaigham Khan
- Huiying Wang
- Krista Zhang

Presenter: Huiying Wang

Take a quick look at the result of
The test set (10,000 new unlabeled images)

How did we train and evaluate our models? - Leave-one-out validation

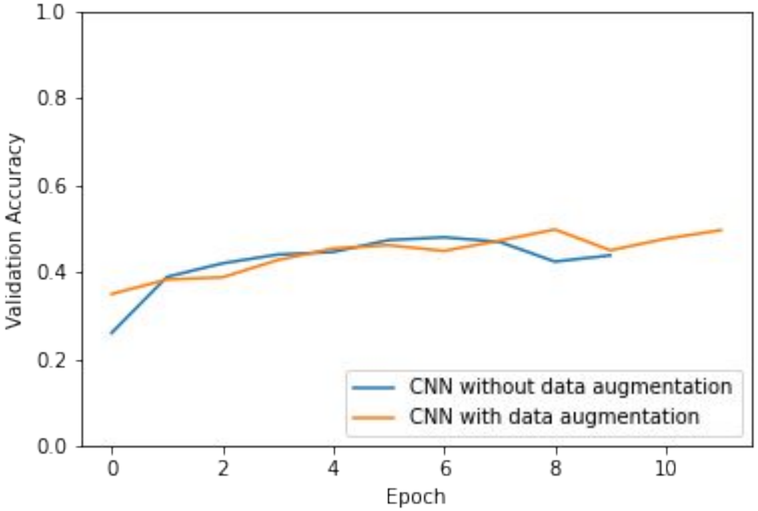
Data we have:

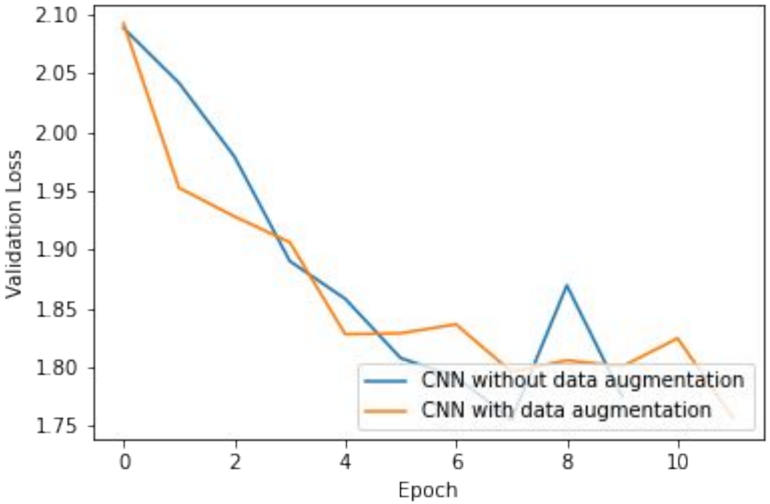
1) Imgs[0:50,000]
2) Noisy_labels[0:50,000]
3) Clean_labels[0:50,000]
4) Cleaned_labels[0:50,000]

		Validation set	Training set
Model I	**CNN without data augmentation**	Imgs[0:10,000], clean_labels	Imgs[10000:50,000], noisy_labels[10000:50,000]
	CNN with data augmentation	Imgs[0:10,000], clean_labels	Imgs[10000:50,000], noisy_labels[10000:50,000]
Model II	**Label correction +CNN with data augmentation**	Imgs[0:10,000], clean_labels	Imgs[10000:50,000],cleaned_labels[10000:50,000]

Model I : So we choose CNN with data augmentation

```
-----The Model 1(a) takes 1.84 seconds to run 10k predictions-----
-----The Model 1(b) takes 1.55 seconds to run 10k predictions-----
```





Structure of Model I(a)

```python
# https://www.tensorflow.org/tutorials/images/cnn
cnn = Sequential()
cnn.add(layers.Conv2D(32, (3,3), padding="same", activation="relu", input_shape=(32, 32, 3)))
cnn.add(layers.MaxPooling2D(2, 2))
cnn.add(layers.Conv2D(64, (3,3), padding="same", activation="relu"))
cnn.add(layers.MaxPooling2D(2, 2))
cnn.add(layers.Conv2D(64, (3,3), padding="same", activation="relu"))

# add dense layers on top
cnn.add(layers.Flatten())
cnn.add(layers.Dense(64, activation='relu'))
cnn.add(layers.Dense(10))

# compile the model
cnn.compile(optimizer='adam', loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])
```

Structure of Model I(b)

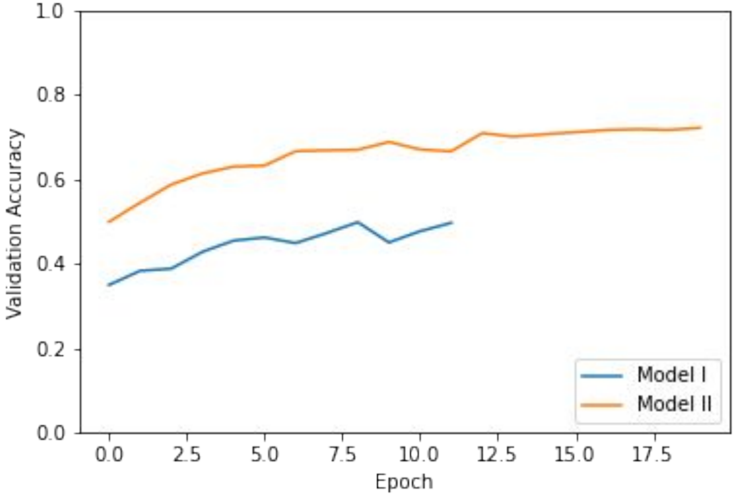
```
# [BUILD A MORE SOPHISTICATED PREDICTIVE MODEL]

# data augomentation
data_augmentation = keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal",
                                                 input_shape=(32,
                                                              32,3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
  ]
)


model_2 = Sequential([
  data_augmentation,
  layers.Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(10)
    ])

# compile the model
model_2.compile(optimizer='adam', loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy
```

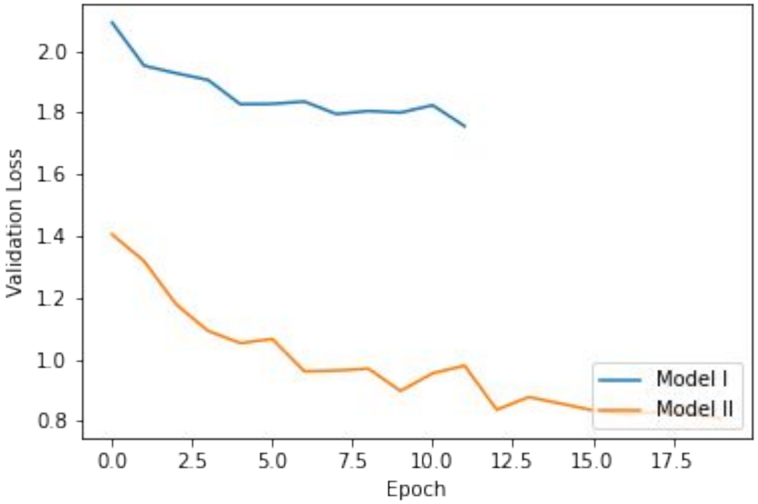
Model II

		Validation set	Training set	input	output
Model II	**Label correction**	Imgs[0:10,000], clean_labels	Imgs[0:10,000], noisy_labels[0:10,000]	noisy_labels[0:50,000]	cleaned_labels[0:50,000]
	CNN with data augmentation	Imgs[0:10,000], clean_labels	Imgs[10000:50,000], cleaned_labels[10000:50,000]		

Model I and Model II: evaluated based on Validation set

```
-----The Model I(b) takes 1.44 seconds to run 10k predictions-----
-----The Model II takes 2.07 seconds to run 10k predictions-----
```





Structure of Label correction model

create a two branch model where one branch consists of feature extractor layers and another one is simply one or more dense layers on top of each other

25]:
```
# branch 1: feature model
img_input = K.Input(shape=(32,32,3))
branch_1 = feature_model
branch_1 = Dense(128, activation="linear")(branch_1.output)

# branch 2: linear
noisy_labels_input = K.Input(shape=(10,))
branch_2 = Dense(128, activation="linear")(noisy_labels_input)
branch_2 = Model(inputs=noisy_labels_input, outputs=branch_2)

# concatenate
concat = concatenate([branch_1, branch_2.output])

# merge two branches
final_model = Dense(64, activation="linear")(concat)
final_model = Dropout(0.25)(final_model)
final_model = Dense(10, activation="softmax")(final_model)

label_correction_model = Model(inputs=[feature_model.input, branch_2.input], outputs=final_model)
```

How is our label correction model?