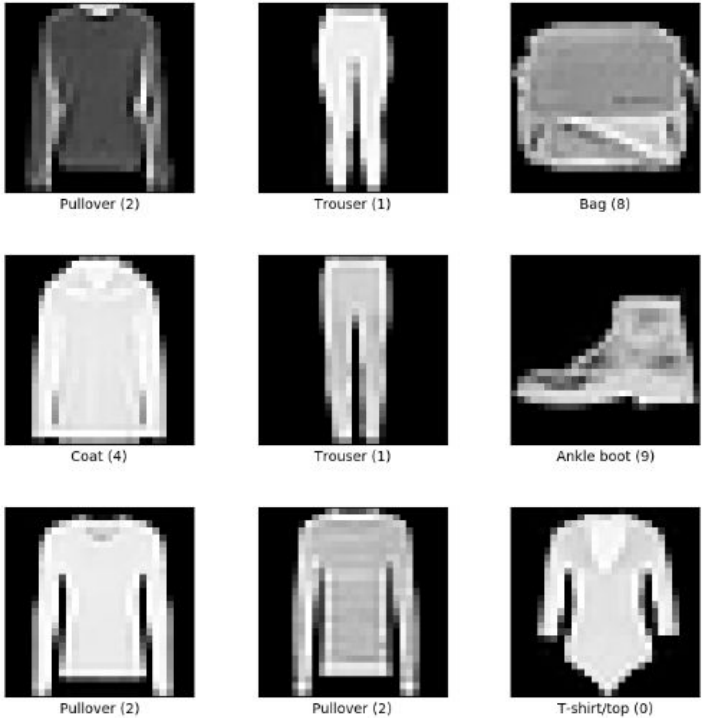
Differential Privacy Applied in Deep Learning

最後一組

M11215032 葉品和 M11215052 陳奕帆 M11215066 鄭宜珊

Dataset : fashion_mnist from keras



https://www.tensorflow.org/datasets/catalog/fashion_mnist

How do you perturb the data?

Dataset : fashion_mnist

```
for idx in range(self._num_microbatches):
    # compute gradient
    microbatch_loss = tf.reduce_mean(tf.gather(microbatches_losses, indices: [idx]))
    grads = gradient_tape.gradient(microbatch_loss, var_list)
    # accountant
    sample_state = self._dp_sum_query.accumulate_record(sample_params, sample_state, grads)

# add noise
grad_sums, self._global_state = (self._dp_sum_query.get_noised_result(sample_state, self._global_state))
```

How do you perturb the data?

```
for idx in range(self._num_microbatches):
    # compute gradient
    microbatch_loss = tf.reduce_mean(tf.gather(microbatches_losses, indices: [idx]))
    grads = gradient_tape.gradient(microbatch_loss, var_list)
    # accountant
    sample_state = self._dp_sum_query.accumulate_record(sample_params, sample_state, grads)

# add noise
grad_sums, self._global_state = (self._dp_sum_query.get_noised_result(sample_state, self._global_state))
```

1

2

3

How do you perturb the data?

```
for idx in range(self._num_microbatches):
    # compute gradient
    microbatch_loss = tf.reduce_mean(tf.gather(microbatches_losses, indices: [idx]))
    grads = gradient_tape.gradient(microbatch_loss, var_list)
    # accountant
    sample_state = self._dp_sum_query.accumulate_record(sample_params, sample_state, grads)

# add noise
grad_sums, self._global_state = (self._dp_sum_query.get_noised_result(sample_state, self._global_state))
```

- define microbatch_loss and then compute gradient of every microbatch by using gradient_tape.gradient ()

How do you perturb the data?

```
for idx in range(self._num_microbatches):
    # compute gradient
    microbatch_loss = tf.reduce_mean(tf.gather(microbatches_losses, indices: [idx]))
    grads = gradient_tape.gradient(microbatch_loss, var_list)
    # accountant
    sample_state = self._dp_sum_query.accumulate_record(sample_params, sample_state, grads)

# add noise
grad_sums, self._global_state = (self._dp_sum_query.get_noised_result(sample_state, self._global_state))
```

- update the set of previous microbatch gradients with the addition of the record argument

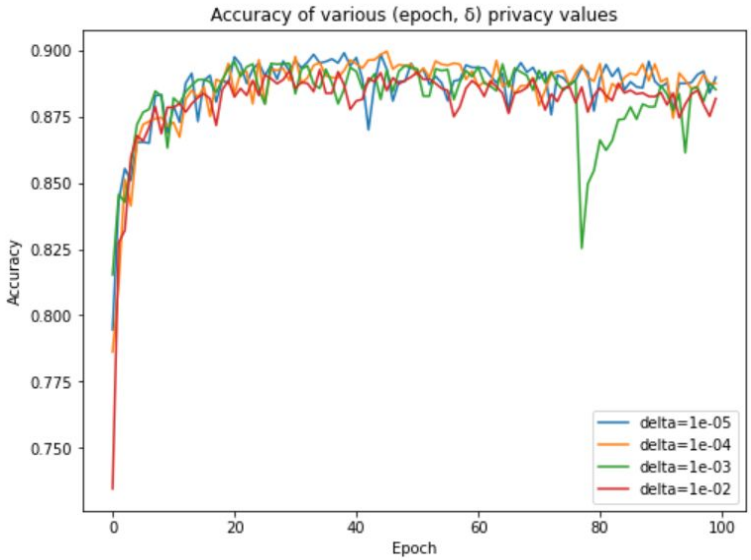
How do you perturb the data?

```
for idx in range(self._num_microbatches):
    # compute gradient
    microbatch_loss = tf.reduce_mean(tf.gather(microbatches_losses, indices: [idx]))
    grads = gradient_tape.gradient(microbatch_loss, var_list)
    # accountant
    sample_state = self._dp_sum_query.accumulate_record(sample_params, sample_state, grads)

# add noise
grad_sums, self._global_state = (self._dp_sum_query.get_noised_result(sample_state, self._global_state))
```

- add noise to the gradient by using get_noised_result ()

Effectiveness measure: Accuracy



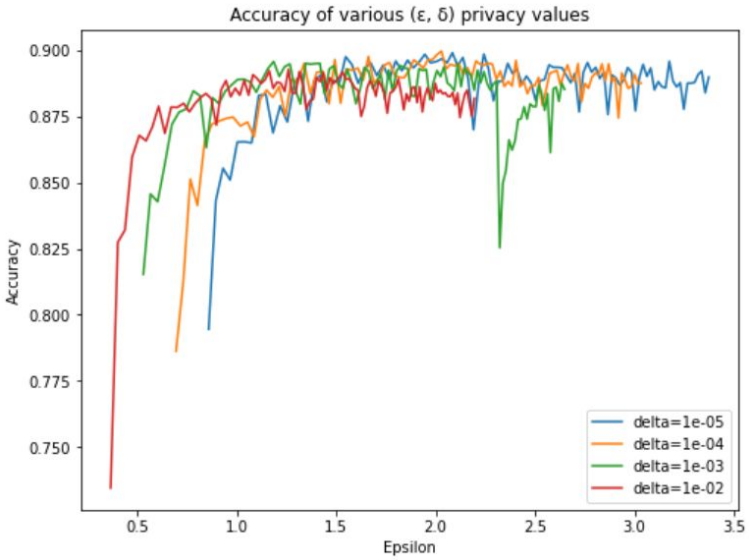
learning rate = 0.001, epoch = 100, C=1.0

σ = 1.2

learning rate = 0.001, epoch = 100, C=1.0

σ = 1.1

Effectiveness measure: Accuracy



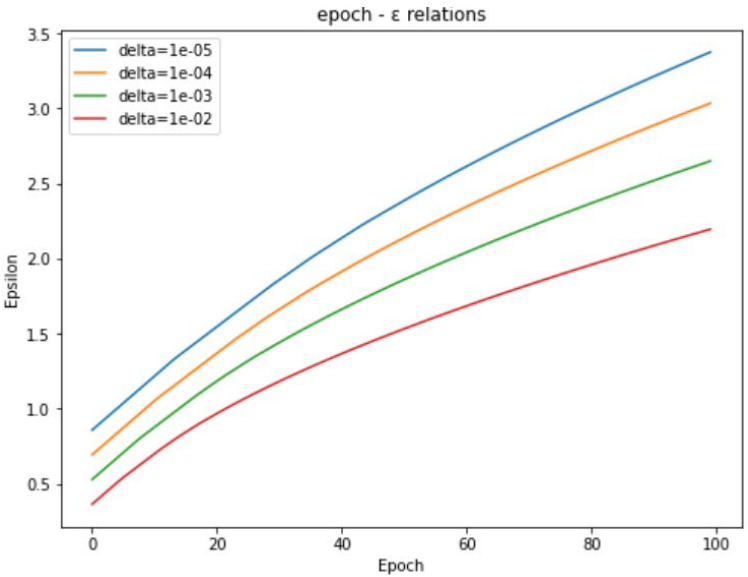
learning rate = 0.001, epoch = 100, C=1.0

σ = 1.2

learning rate = 0.001, epoch = 100, C=1.0

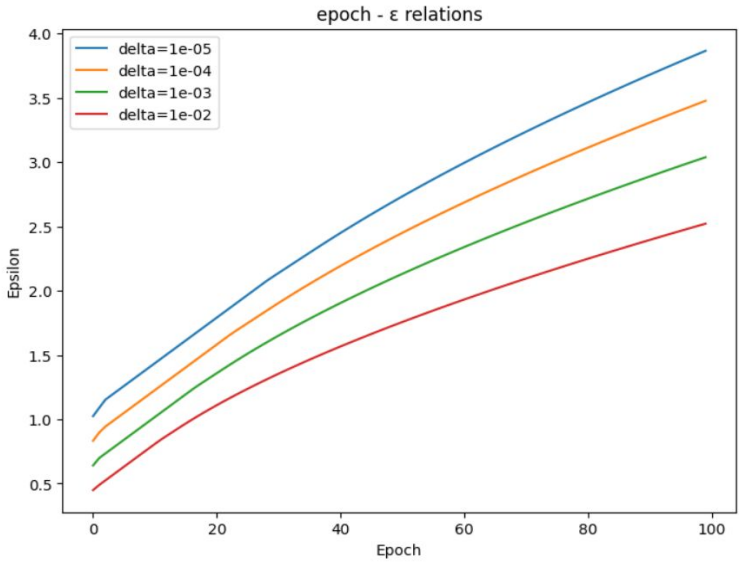
σ = 1.1

Privacy level: the value of ε



learning rate = 0.001, epoch = 100, C=1.0

σ = 1.2



learning rate = 0.001, epoch = 100, C=1.0

σ = 1.1