دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین ۵

علی صداقی

۹۷۵۲۱۳۷۸

سوال ۱) در پاسخ قسمت الف مقالات زیر مطالعه شده است:

https://iopscience.iop.org/article/10.1088/1742-6596/1237/3/032048

https://www.mdpi.com/2076-3417/10/7/2253/pdf

https://arxiv.org/pdf/2005.08649.pdf

در واقع می‌توان این مسئله را یک مسئله Localization در نظر گرفت که جنس متفاوتی با مسائل Classification و Regression دارد. به همین دلیل تابع فعالسازی و ضرر آن نیز کمی متفاوت خواهد بود.

الف) با توجه به اینکه به دنبال ۵ نقطه هستیم و هر نقطه دارای دو مولفه x و y است پس در لایه خروجی از ۱۰ نورون استفاده می‌کنیم. با توجه به اینکه هر ۲ نورون در این لایه وابستگی معنایی به هم دارند (تشکیل یک نقطه) باید آن ها را در قالب ۵ دسته ۲ تایی ببینیم.

مثلا اگر به طور ساده لوحانه هیچ تابع فعال‌سازی استفاده نکنیم و تنها Logits ها را در نظر بگیریم و از تابع ضرر MSE استفاده کنیم (Linear Regression) عملا مختصات‌های x و y را بدون توجه به همدیگر بهینه می‌کنیم و نتیجه به‌هیچ عنوان قابل قبول نخواهد بود. پس این دسته‌بندی الزامی می‌باشد.

می‌توان مختصات نقاط در داده آموزشی را بین 0 تا 1 Scale کنیم (تقسیم کردن بر Height و Width تصاویر) سپس در لایه خروجی مدل از تابع Sigmoid استفاده کنیم تا مقادیر خروجی و مقادیر واقعی در یک بازه باشند. همانطور که در کلاس درس هم بررسی شد یکی از کاربردهای دیگر تابع Sigmoid مقیاس کردن مقادیر بین 0 تا 1 می‌باشد (جدای توزیع احتمالی).

می‌توان این Scale کردن را انجام نداد و اجازه داد نورون‌های لایه آخر هر مقدار دلخواهی را بگیرند. در واقع از هیچ تابع فعال‌سازی استفاده نکرد و همان مقادیر Logits را برگردانیم.

- ✓ به طور خلاصه: لایه خروجی باید شامل $2 \times Num\ of\ Landmarks$ نورون باشد و می‌توانید تابع فعالسازی Sigmoid یا Tanh (در صورت Scale کردن) یا بدون تابع فعالسازی باشد (بدون Scale)

تابع ضرر: در تمامی مقالات بالا از تابع ضرری مبتنی بر فاصله نقاط استفاده شده و تفاوت ها در ثابت فرمول ها می‌باشد. توابع ضرر بکار رفته در مقالات به ترتیب به صورت زیر است:

$$err = \sqrt{(x-x')^2 + (y-y')^2} \,/\, l,$$

$$N\_Loss(p,\hat{p}) = \frac{1}{68}\sum_{i=1}^{68} \frac{\alpha\left|(x_i - \hat{x}_i)\right|}{W} + \frac{\beta\left|(y_i - \hat{y}_i)\right|}{H}$$

در این حالت Scale کردن را نداریم.

To train a regression approach, the L2 distance is adopted to evaluate the point-wise difference between the detected and the ground-truth landmarks, which can be formulated as

$$loss_{reg} = \frac{1}{|L|}\sum_{l \in L} \|S_l - \hat{S}_l\|_2, \qquad (1)$$

- ✓ به طور خلاصه: تابع ضرر باید بر پایه فاصله نقاط باشد و می‌تواند فاصله L1 یا L2 را در نظر بگیرد (L2 مناسب تر است). همچنین با توجه به تابع فعالسازی و Scale کردن یا نکردن می‌تواند ثابت‌های متفاوتی را نیز داشته باشد.
- ✓ در مقاله سوم دو روش Heatmap Approaches و A PIXEL-WISE CLASSIFICATION MODEL WITH DISCRIMINATOR نیز ارائه شده که از سطح بحث ما بسیار پیچیده تر می‌باشد.

ب) در این کد به جای ۵ نقطه به دنبال ۷۶ نقطه Landmark هستیم. از شبکه ResNet50 که داخل Keras موجود است به عنوان مدل پایه (Base Model) استفاده شده است که وزن‌های آن از آموزش دیتاست ImageNet بدست آمده است. جهت کاهش ابعاد خروجی شبکه ResNet50 یک Global Average Pooling به آن اضافه شده است. لایه خروجی (Output): با توجه به اینکه می‌خواهیم ۷۶ نقطه Landmark پیدا کنیم از یک لایه با ۱۵۲ نورون (به ازای هر نقطه ۲ نورون یکی برای مختصات x و دیگری y) استفاده شده است.

تابع فعال‌سازی هر نورون در لایه خروجی Sigmoid می‌باشد. بنابراین هر ۲ نورون در این لایه مختصات x و y یک نقطه Landmark را به دست می‌آورند که مقدار آن بین ۰ تا ۱ است.

```
X = Input(shape=IMAGE_SHAPE)

baseModel = ResNet50(include_top=False)

pooled = GlobalAveragePooling2D()(baseModel(X))

dense = Dense(2 * NUM_LANDMARKS, activation="sigmoid", …)

out = dense(pooled)

model = Model(inputs=X, outputs=out)
```

در پایان برای اینکه مختصات بین ۰ و ۱ به دست آمده، قابل نمایش بر روی تصویر اصلی باشد هر یک را در WIDTH و HEIGHT ورودی ضرب می‌کنیم.

```
points = model.predict(test_samples[j:j+1])[0]

point[0] = point[0]*image_shape[1]

point[1] = point[1]*image_shape[0]
```

تابع ضرر: با توجه به اینکه می‌خواهیم نقاطی شامل x و y را بیابیم و با نقاط واقعی مقایسه کنیم بهتر است از یک تابع ضرری استفاده کنیم که مبتنی بر فاصله نقاط است. در این پیاده سازی از فاصله L2 یا فاصله اقلیدسی استفاده شده است. برای انجام این کار ابتدا ۱۵۶ نورون و ۱۵۶ عدد واقعی را تغییر شکل (Reshape) می‌دهیم تا به ۷۶ دسته دوتایی تبدیل شود. (همگی مقادیری بین ۰ تا ۱ دارند.)

```
T_reshaped = tf.reshape(T, shape=[-1, 2, NUM_LANDMARKS])

Y_reshaped = tf.reshape(Y, shape=[-1, 2, NUM_LANDMARKS])
```

سپس فاصله اقلیدسی را بین نقاط پیشبینی شده (Y_reshaped) و نقاط واقعی (T_reshaped) حساب می‌کنیم.

```
distance = tf.norm(T_reshaped - Y_reshaped, ord='euclidean', axis=1)
```

با توجه به اینکه ممکن است در این دیتاست تعدادی از Landmark ها مشخص نشده باشند (نقطه 0,0 تخصیص یافته) در محاسبه فاصله اقلیدسی نباید آن هارا در نظر بگیریم زیرا باعث می‌شود شبکه اطلاعات غلطی را آموزش ببینید. پس با ایجاد یک Mask از مقادیر واقعی اثر این نقاط (0,0) را از بین می‌بریم.

```
#If summing x and y yields zero, then (x, y) == (0, 0)

T_summed = tf.reduce_sum(T_reshaped, axis=1)

zero = tf.constant(0.0)

mask = tf.not_equal(T_summed, zero)

#Get the interested samples and calculate loss with Mean of Euclidean distance

masked_distance = tf.boolean_mask(distance, mask

return tf.reduce_mean(masked_distance)
```

سوال ۲) اگر می‌خواستیم قیمت گوشی را تخمین بزنیم و یک مقدار پیوسته بدست بیاوریم (Regression) بهترین انتخاب به صورت زیر بود:

لایه خروجی: یک نورون بدون تابع فعال‌سازی

تابع ضرر: MSE

اما این مسئله قیمت را به صورت دقیق خروجی نمی‌دهد بلکه برای قیمت ۴ رنج (Class) دارد پس یک مسئله Classification است. بنابراین باید به صورت زیر عمل کنیم:

لایه خروجی: ۴ نورون با تابع فعالسازی Softmax

تابع ضرر: Categorical Cross entropy

این دیتاست شامل ۲۰۰۰ رکورد می‌باشد که هر یک دارای ۲۰ ویژگی است و خروجی آن نیز یکی از ۴ کلاس 0، 1، 2، 3 می‌باشد.

۲۰ ویژگی را به صورت زیر نرمالیزه می‌کنیم:

```
x -= np.mean(x)
x /= np.std(x)
```

لایه آخر (خروجی) را به صورت زیر تعریف می‌کنیم:

```
### YOU HAVE TO MAKE YOUR CHANGES HERE !!!
model.add(Dropout(0.2))
model.add(Dense(4, activation="softmax"))
```

تابع ضرر و بهینه‌ساز را به صورت زیر تعریف می‌کنیم:

```
LOSS = 'categorical_crossentropy'
OPTIMIZER = SGD(
  learning_rate=0.001,
  momentum=0.9,
  nesterov=True,
)
```

با استفاده از Model Checkpoint بهترین شبکه با توجه به مقدار Validation Accuracy را ذخیره می‌کنیم:

```
checkpoint = ModelCheckpoint(
  filepath="model.hdf5",
  monitor='val_accuracy',
  save_best_only=True,
  save_weights_only=False,
  mode='auto')
```

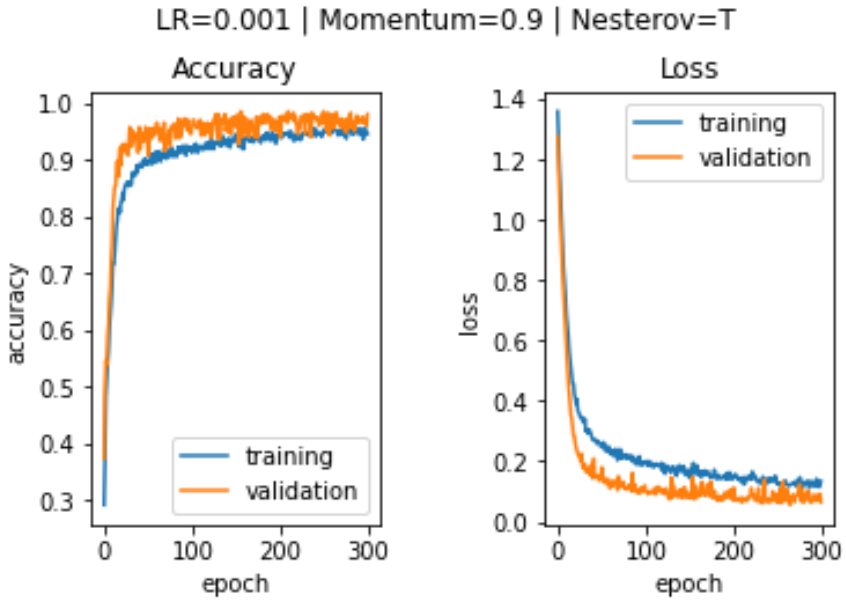
ساختار مدل:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_51 (Dense)             (None, 300)               6300
_________________________________________________________________
dropout_40 (Dropout)         (None, 300)               0
_________________________________________________________________
dense_52 (Dense)             (None, 200)               60200
_________________________________________________________________
dropout_41 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_53 (Dense)             (None, 150)               30150
_________________________________________________________________
dropout_42 (Dropout)         (None, 150)               0
_________________________________________________________________
dense_54 (Dense)             (None, 50)                7550
_________________________________________________________________
dropout_43 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_55 (Dense)             (None, 4)                 204
=================================================================
Total params: 104,404
Trainable params: 104,404
Non-trainable params: 0
```

نتایج:

```
Train Result
50/50 [==============================] - 0s 1ms/step - loss: 0.0617 -
accuracy: 0.9775

Test Result
13/13 [==============================] - 0s 3ms/step - loss: 0.0610 -
accuracy: 0.9900
```



سوال ۳) با یک مسئله Classification رو به رو هستیم پس باید انتظار داشته باشیم که تابع فعالسازی Softmax و تابع ضرر Categorical Cross Entropy بهترین نتیجه را داشته باشد.

```
NUM_WORDS = 1000
CLASSES = 46
EPOCHS = 20
BATCH_SIZE = 512
```

نتایج:

Config	Train	Validation
Sigmoid | BCE	loss: 0.6547 accuracy: 0.0198	loss: 0.6550 accuracy: 0.0196
Softmax | CCE	loss: 1.9464 accuracy: 0.5173	loss: 0.5173 accuracy: 0.5312
Sigmoid | MSE	loss: 0.2509 accuracy: 0.0183	loss: 0.2509 accuracy: 0.0245
Softmax | MSE	loss: 0.0213 accuracy: 0.0199	loss: 0.0213 accuracy: 0.0214
Linear | MSE	loss: 0.0295 accuracy: 0.0982	loss: 0.0292 accuracy: 0.1060

✓ با توجه به اینکه هر تابع ضرر فرمولی مخصوص خود را دارد نمی‌توان ملاک و Metric مقایسه را مقدار loss قرار داد و ملاک Accuracy می‌باشد زیر فرمول آن برای تمامی حالات یکسان است.

ترتیب Accuracy در Train:

Softmax | CCE > Linear | MSE > Softmax | MSE > Sigmoid | BCE > Sigmoid | MSE

ترتیب Accuracy در Validation:

Softmax | CCE > Sigmoid | MSE > Softmax | MSE > Sigmoid | BCE > Linear | MSE

مقایسه: با توجه به توضیحاتی که در ابتدای پاسخ نیز داده شد Softmax | CCE قطعا بهترین نتیجه را دارد (Multi Class و Single Label). نمودارهای آن نیز نوسان بسیار کمی دارند.

بدترین حالت: با توجه به اینکه در ۲۰ ایپاک نمیتوانستیم با اعتماد زیادی صحبت کنیم الگوریتم را برای ۱۰۰ ایپاک اجرا کردیم و نتایج آن را در زیر آوردیم. با توجه به نتایج حالت تابع فعالسازی Softmax و تابع ضرر MSE بدترین نتیجه را دارد. دلیل آن این است که در این حالت مشتق تابع ضرر همواره نزدیک 0 خواهد بود و وزن‌ها آپدیت نخواهد شد (Gradient Vanishing). در واقع با رویکرد Maximum Likelihood اگر مسئله را بررسی کنیم تابع ضرر MSE برای تابع فعالسازی Sigmoid اصلا مناسب نیست. نمودار خطای آن نیز عملا یک خط با شیب ثابت شده است که مقدار شیب آن تقریبا نزدیک 0 است و عملا بهینه‌سازی نداریم.

✓ اگر تعداد Epoch را بجای ۲۰ عدد ۱۰۰ در نظر بگیریم نتایج زیر به دست خواهد آمد:

Config	Train	Validation
Sigmoid | BCE	accuracy: 0.3678	accuracy: 0.3882
Softmax | CCE	accuracy: 0.6964	accuracy: 0.6986
Sigmoid | MSE	accuracy: 0.1939	accuracy: 0.1999
Softmax | MSE	accuracy: 0.0046	accuracy: 0.0062
Linear | MSE	accuracy: 0.3862	accuracy: 0.3887

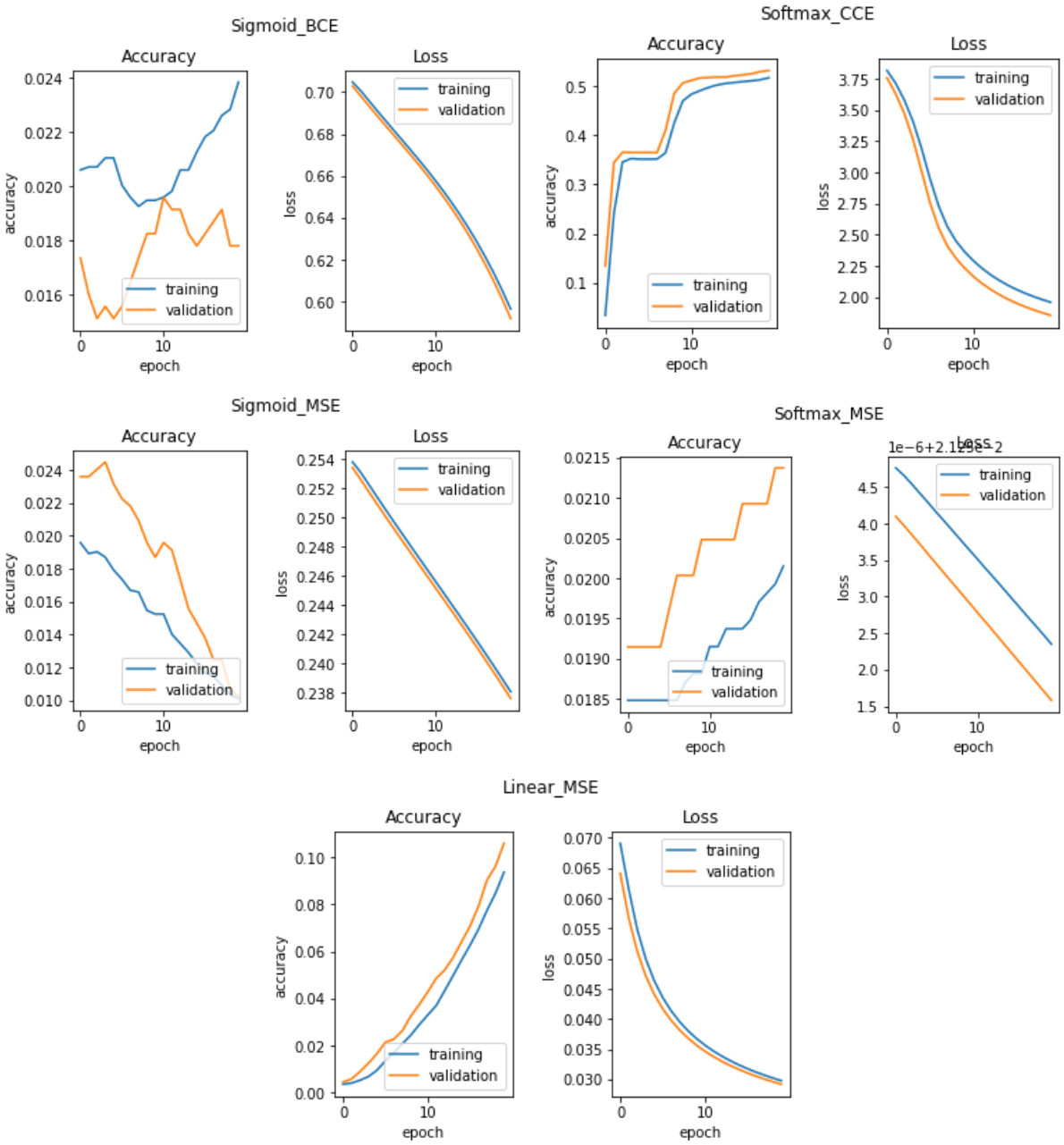
ترتیب Accuracy در Train:

Softmax | CCE > Linear | MSE > Sigmoid | BCE > Sigmoid | MSE > Softmax | MSE

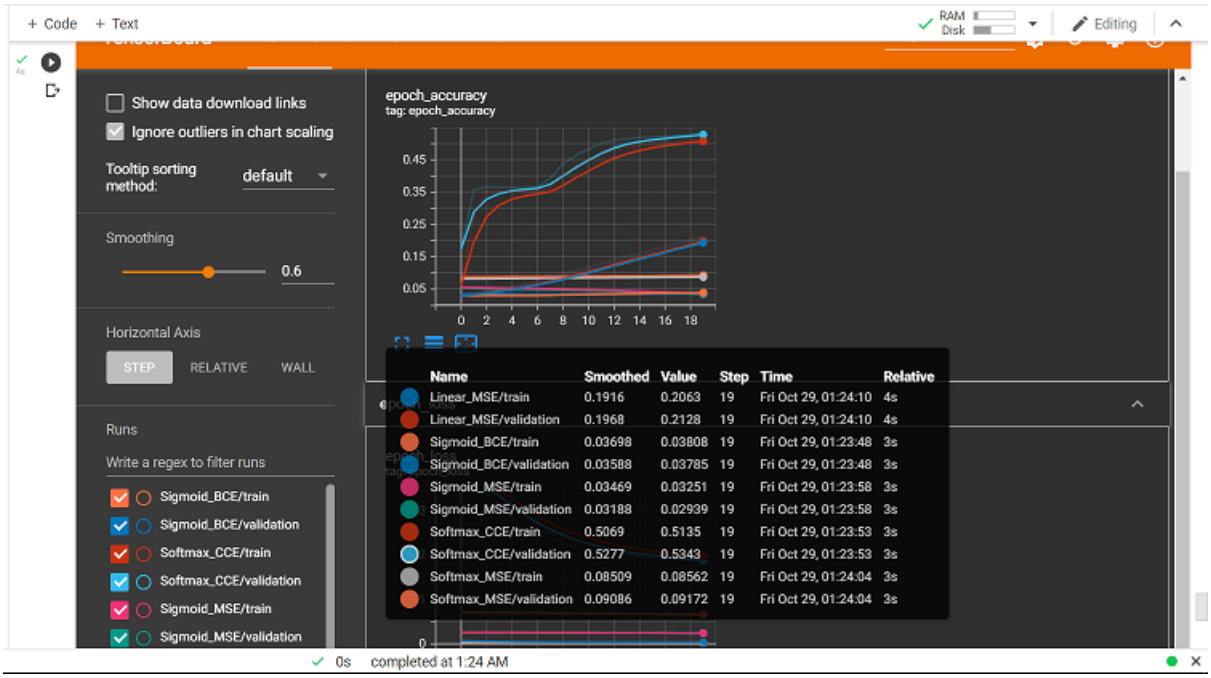
ترتیب Accuracy در Validation:

Softmax | CCE > Linear | MSE > Sigmoid | BCE > Sigmoid | MSE > Softmax | MSE

نمودار Loss و Accuracy با 20 اپاک:



✓ نتایج Tensor board در نوتبوک و صفحه بعدی موجود است.


+ Code
+ Text
RAM
Disk
Editing
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method:
default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
Sigmoid_BCE/train
Sigmoid_BCE/validation
Softmax_CCE/train
Softmax_CCE/validation
Sigmoid_MSE/train
Sigmoid_MSE/validation
epoch_accuracy
tag: epoch_accuracy
0.45
0.35
0.25
0.15
0.05
0 2 4 6 8 10 12 14 16 18
Name Smoothed Value Step Time Relative
Linear_MSE/train 0.1916 0.2063 19 Fri Oct 29, 01:24:10 4s
Linear_MSE/validation 0.1968 0.2128 19 Fri Oct 29, 01:24:10 4s
Sigmoid_BCE/train 0.03698 0.03808 19 Fri Oct 29, 01:23:48 3s
Sigmoid_BCE/validation 0.03588 0.03785 19 Fri Oct 29, 01:23:48 3s
Sigmoid_MSE/train 0.03469 0.03251 19 Fri Oct 29, 01:23:58 3s
Sigmoid_MSE/validation 0.03188 0.02939 19 Fri Oct 29, 01:23:58 3s
Softmax_CCE/train 0.5069 0.5135 19 Fri Oct 29, 01:23:53 3s
Softmax_CCE/validation 0.5277 0.5343 19 Fri Oct 29, 01:23:53 3s
Softmax_MSE/train 0.08509 0.08562 19 Fri Oct 29, 01:24:04 3s
Softmax_MSE/validation 0.09086 0.09172 19 Fri Oct 29, 01:24:04 3s
0
0s
completed at 1:24 AM