深度学习第一次作业

未央-能动 11　张立博　2021012487

2024 年 10 月 31 日

1 Backpropagation

1.1

softmax 函数的定义如下：

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

令 output:

$$y_i = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

当 $i = k$ 时:

$$\begin{aligned}\frac{\partial y_i}{\partial z_k} &= \frac{e^{z_i}\sum_{j=1}^{K} e^{z_j} - e^{z_i}\cdot e^{z_i}}{(\sum_{j=1}^{K} e^{z_j})^2} \\ &= y_i(1-y_i)\end{aligned}$$

当 $i \neq k$ 时:

$$\begin{aligned}\frac{\partial y_i}{\partial z_k} &= -\frac{e^{z_i}\cdot e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \\ &= -y_i y_k\end{aligned}$$

综上:

$$\frac{\partial y_i}{\partial z_k} = \begin{cases} y_i(1-y_i), & \text{如果} i = k \\ -y_i y_k, & \text{如果} i \neq k \end{cases}$$

转化为更紧凑的矩阵形式:

$$\frac{\partial Y}{\partial Z} = \text{diag}(Y) - Y \cdot Y^T$$

1.2 Feed-forward computations

对于 $X_i \in \mathbb{R}^{L\times D}$, feed-forward 过程如下:

1. Transpose: $X_i^T \in \mathbb{R}^{D\times L}$

2. FC1:

$$Z_1 = \mathrm{ReLU}(X_i^T \cdot \Theta_1 + b_1) \in \mathbb{R}^{D\times L}$$

其中 $\Theta_1 \in \mathbb{R}^{L\times L}, b_1 \in \mathbb{R}^L$

3. Transpose: $Z_1^T \in \mathbb{R}^{L\times D}$

4. Element-wise addition:

$$Z = Z_1^T + X_i \in \mathbb{R}^{L\times D}$$

5. FC2:

$$Z_2 = \mathrm{ReLU}(Z \cdot \Theta_2 + b_2) \in \mathbb{R}^{L\times D}$$

其中 $\Theta_2 \in \mathbb{R}^{D\times D}, b_2 \in \mathbb{R}^D$

6. Mean:

$$Z_{\text{mean}}(i) = \frac{1}{D}\sum_{j=1}^{D} Z_2(i,j) \in \mathbb{R}^L, i = 0, 1, \cdots, L-1$$

7. FC3:

$$Z_3 = \mathrm{Softmax}(Z_{\text{mean}} \cdot \Theta_3 + b_3) \in \mathbb{R}^K$$

其中 $\Theta_3 \in \mathbb{R}^{L\times K}, b_3 \in \mathbb{R}^K$

8. Output:

$$\hat{Y}_i = Z_3$$

$\hat{Y} = [\hat{Y}_1, \hat{Y}_2, \cdots, \hat{Y}_m] \in \mathbb{R}^{m\times K}$

1.3 Compute the gradients

已知

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}[-\sum_{k=1}^{K} Y_k^i \log(\hat{Y}_k^i)]$$

则

$$\frac{\partial \mathcal{L}}{\partial \hat{Y}^i} = -\frac{Y^i}{\hat{Y}^i}$$

对于 FC3 层, 令:

$$
\begin{aligned}
T_3 &= Z_{\text{mean}} \cdot \Theta_3 + b_3 \\
\hat{Y}^i &= Z_3 = \text{Softmax}(T_3)
\end{aligned}
$$

则

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \Theta_3} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{Y}^i}\frac{\partial \hat{Y}^i}{\partial T_3}\frac{\partial T_3}{\partial \Theta_3} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{Y^i}{\hat{Y}^i}(\text{diag}(\hat{Y}^i) - \hat{Y}^i(\hat{Y}^i)^T)Z_{\text{mean}}^T \in \mathbb{R}^{L\times K}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_3} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{Y}^i}\frac{\partial \hat{Y}^i}{\partial T_3}\frac{\partial T_3}{\partial b_3} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{Y^i}{\hat{Y}^i}(\text{diag}(\hat{Y}^i) - \hat{Y}^i(\hat{Y}^i)^T) \in \mathbb{R}^{K}
\end{aligned}
$$

对于 FC2 层, 令:

$$
\begin{aligned}
T_2 &= Z \cdot \Theta_2 + b_2 \\
Z_2 &= \text{ReLU}(T_2)
\end{aligned}
$$

则

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \Theta_2} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{Y}^i}\frac{\partial \hat{Y}^i}{\partial T_3}\frac{\partial T_3}{\partial Z_{\text{mean}}}\frac{\partial Z_{\text{mean}}}{\partial Z_2}\frac{\partial Z_2}{\partial T_2}\frac{\partial T_2}{\partial \Theta_2} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{Y^i}{\hat{Y}^i}(\text{diag}(\hat{Y}^i) - \hat{Y}^i(\hat{Y}^i)^T)\Theta_3^T\frac{\partial Z_{\text{mean}}}{\partial Z_2}\text{ReLU}'(T_2)Z^T \in \mathbb{R}^{D\times D}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_2} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{Y}^i}\frac{\partial \hat{Y}^i}{\partial T_3}\frac{\partial T_3}{\partial Z_{\text{mean}}}\frac{\partial Z_{\text{mean}}}{\partial Z_2}\frac{\partial Z_2}{\partial T_2}\frac{\partial T_2}{\partial b_2} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\frac{Y^i}{\hat{Y}^i}(\text{diag}(\hat{Y}^i) - \hat{Y}^i(\hat{Y}^i)^T)\Theta_3^T\frac{\partial Z_{\text{mean}}}{\partial Z_2}\text{ReLU}'(T_2) \in \mathbb{R}^{D}
\end{aligned}
$$

其中 $\partial Z_{\text{mean}}/\partial Z_2$ 是平均操作的导数，$\text{ReLU}'(T_2)$ 是 ReLU 函数的导数。
对于 FC1 层, 令:

$$
\begin{aligned}
T_1 &= X_i^T \cdot \Theta_1 + b_1 \\
Z_1 &= \text{ReLU}(T_1)
\end{aligned}
$$

定义 Residual:

$$
\begin{aligned}
\delta_2 &= \frac{\partial \mathcal{L}}{\partial Z_2} \\
&= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial \hat{Y}^i}\frac{\partial \hat{Y}^i}{\partial T_3}\frac{\partial T_3}{\partial Z_{\text{mean}}}\frac{\partial Z_{\text{mean}}}{\partial Z_2} \\
&= -\frac{1}{m}\sum_{i=1}^{m} \frac{Y^i}{\hat{Y}^i}(\text{diag}(\hat{Y}^i) - \hat{Y}^i(\hat{Y}^i)^T)\Theta_3^T \frac{\partial Z_{\text{mean}}}{\partial Z_2}
\end{aligned}
$$

则

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \Theta_1} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial Z_2}\frac{\partial Z_2}{\partial T_2}\frac{\partial T_2}{\partial Z}\frac{\partial Z}{\partial Z_1}\frac{\partial Z_1}{\partial T_1}\frac{\partial T_1}{\partial \Theta_1} \\
&= -\frac{1}{m}\sum_{i=1}^{m} \delta_2 \text{ReLU}'(T_2)\Theta_2^T \text{ReLU}'(T_1)X_i^T \in \mathbb{R}^{L\times L}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial b_1} &= \frac{1}{m}\sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial Z_2}\frac{\partial Z_2}{\partial T_2}\frac{\partial T_2}{\partial Z}\frac{\partial Z}{\partial Z_1}\frac{\partial Z_1}{\partial T_1}\frac{\partial T_1}{\partial b_1} \\
&= -\frac{1}{m}\sum_{i=1}^{m} \delta_2 \text{ReLU}'(T_2)\Theta_2^T \text{ReLU}'(T_1) \in \mathbb{R}^{L}
\end{aligned}
$$

1.4 Pseudo-code for SGD

Algorithm 1 Pseudo-code for SGD

```
for i = 1 to num_epochs do
    for j = 1 to num_batches do
        (X, Y) = get_batch(j)                ▷ randomly select a batch of data
        Ŷ = feed_forward(X)                  ▷ feed-forward computation
        L(Θ_t) = loss(Y, Ŷ)                  ▷ compute loss
        Δ_t = ∇_Θ L(Θ_t), t = 1, 2, 3        ▷ compute gradients
        Θ_{t+1} = Θ_t − ηΔ_t                 ▷ update parameters
    end for
end for
```

2 MLP

2.1 Implement and Visualization

2.2 Implement

完整实现见 `MLP/model.py`

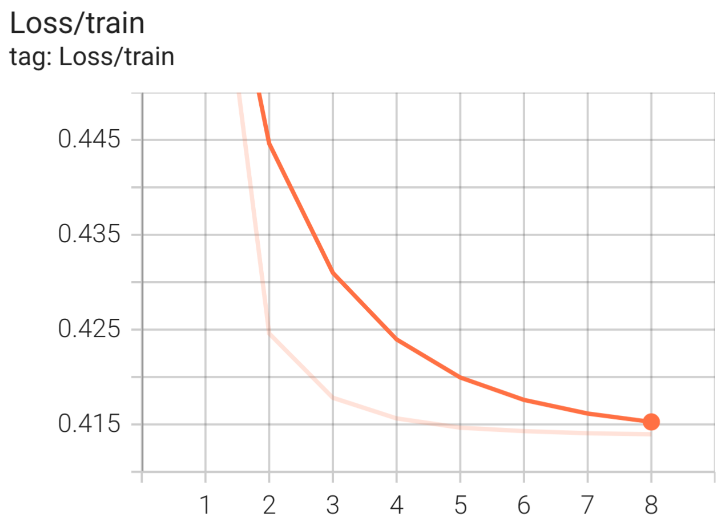
对于权重初始化, trend and seasonal component 前后两个 mlp 分别使用 He 初始化和随

机初始化

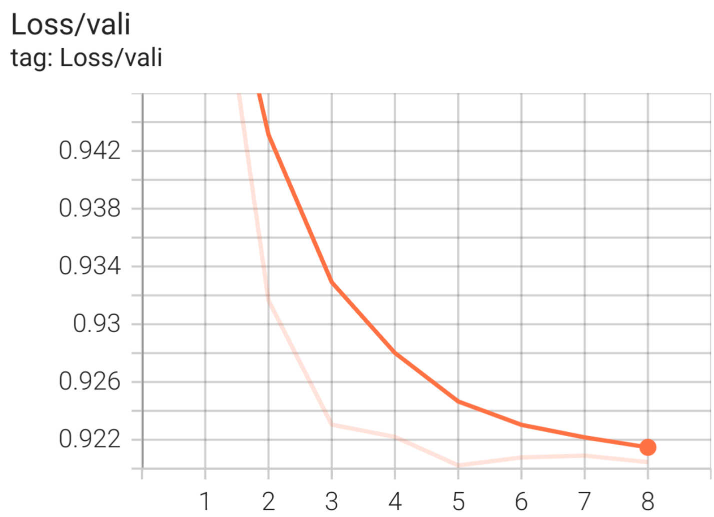
2.3 Visualization

使用默认超参数进行训练，训练过程中各损失变化如下:

1. train loss

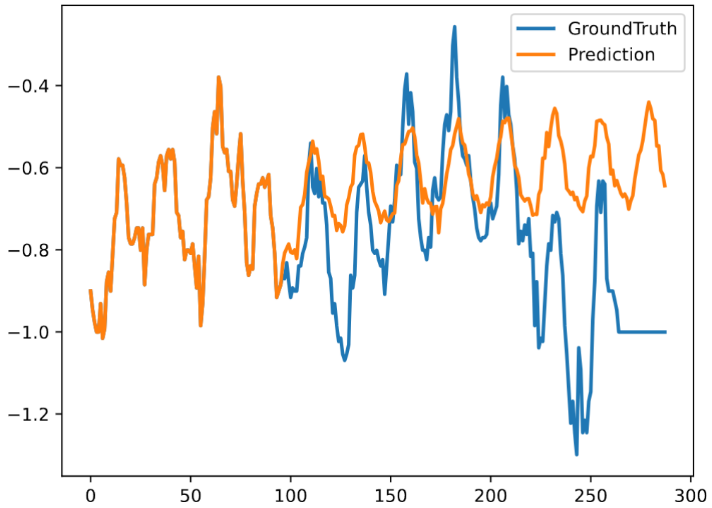


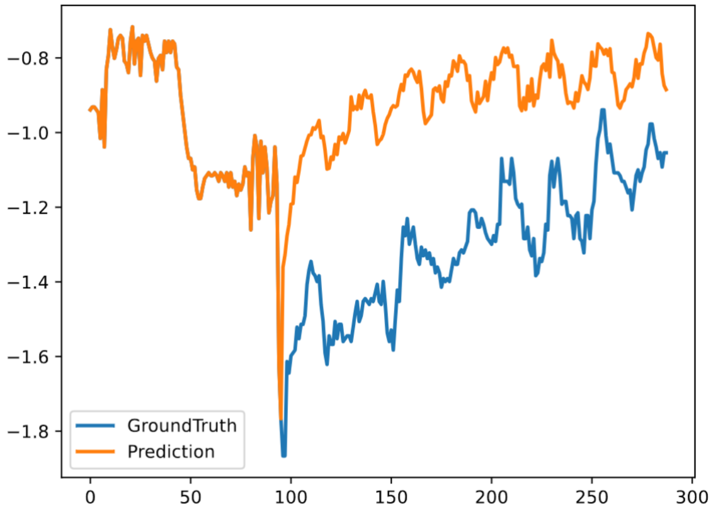
2. valid loss

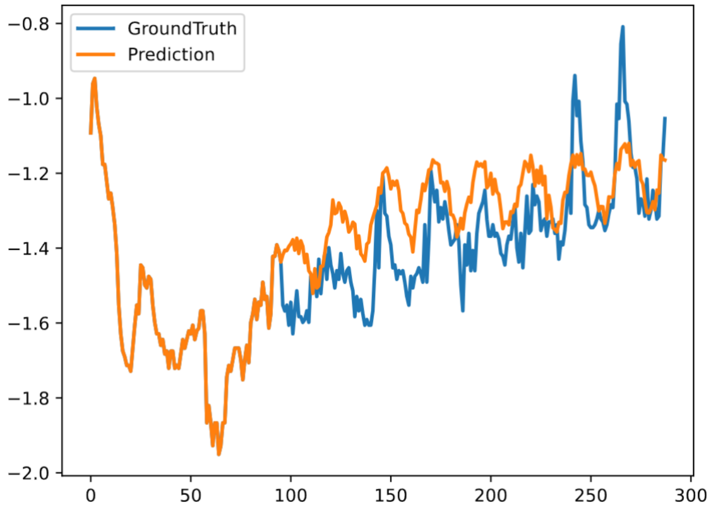


上述损失曲线为平滑后的结果 (平滑因子为 0.6)

模型的一些预测结果如下:


GroundTruth
Prediction


GroundTruth
Prediction


GroundTruth
Prediction

2.4 Train using different hyper-parameters

分别在改变 learning rate, hidden size, 其余超参数为默认参数的 setting 下进行训练, 结果如下:

1. change learning rate

learning rate	MSE	MAE
0.05	**0.4548**	**0.4467**
0.01	0.4926	0.4757
0.001	0.7194	0.6072

表 1: Change learning rate

2. change hidden size

hidden size	MSE	MAE
512	0.4548	0.4467
1024	**0.4528**	**0.4429**
2048	0.4550	0.4435

表 2: Change hidden size

实验发现，改变 learning rate 和 hidden size 对模型的性能影响较小，适当增加 hidden size 可以提高模型的性能

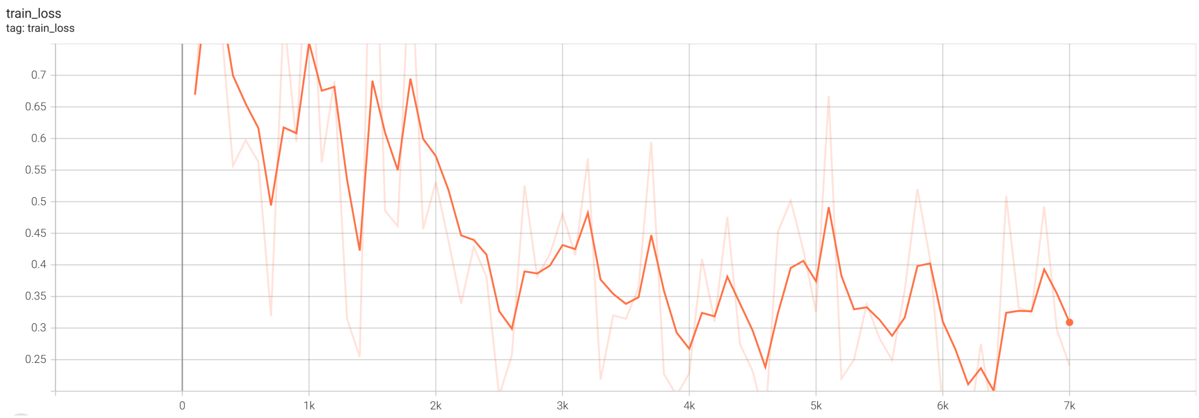
在该数据集上进行的所有实验中，当 learning rate=0.05, hidden size=1024 时，模型性能最好

3 CNN

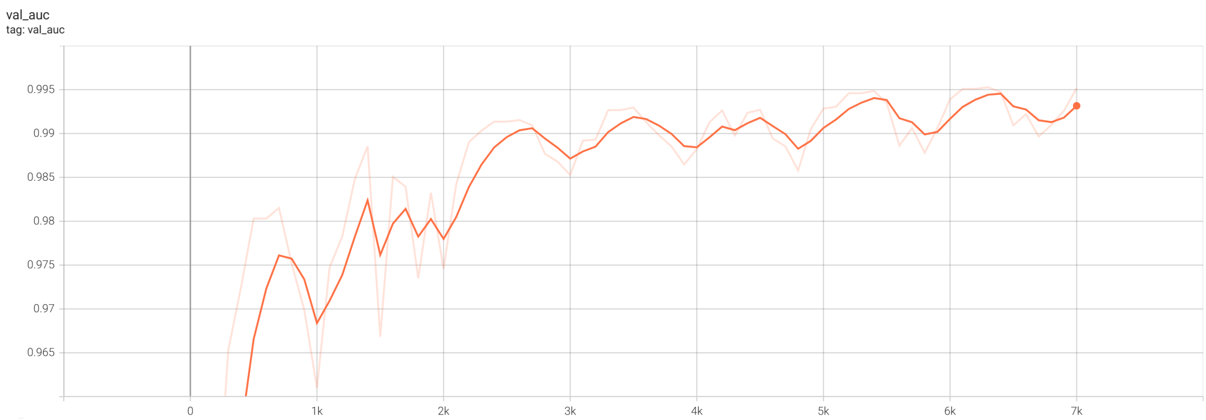
3.1 Train Model A

从零开始训练 ResNet-18, 用 TensorBoard 记录训练过程中的 train loss, valid loss (每 100 次迭代记录一次), 结果如下:

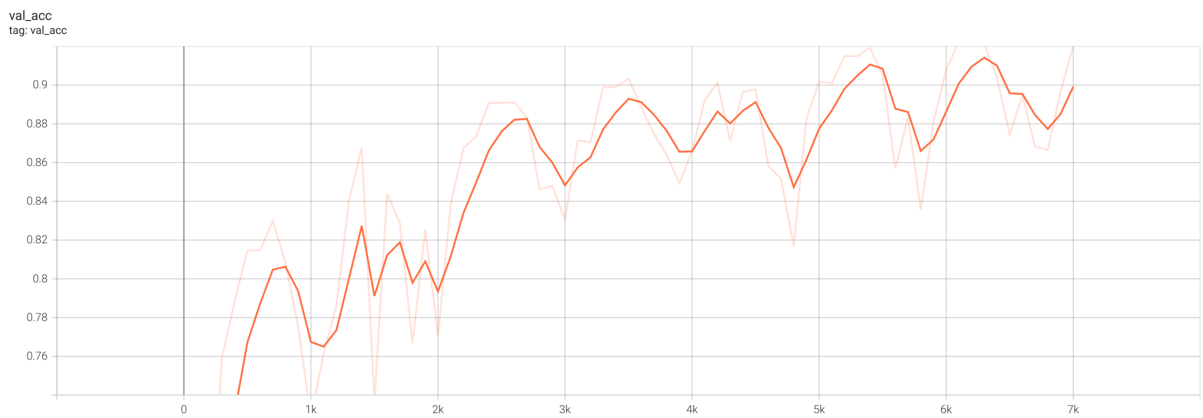
1. train loss



2. valid auc score



3. valid acc score



测试结果 auc:0.982,acc:0.868

3.2 Visualize conv features

可视化 ResNet-18 中间层的卷积特征, 每一层特征图选择其中一个通道, 结果如下:

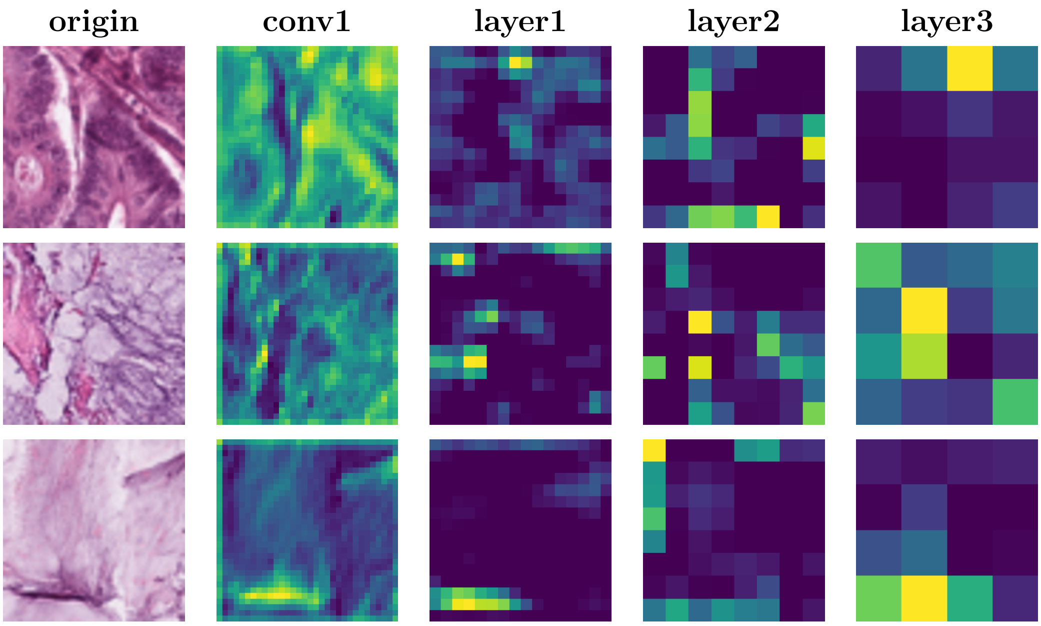


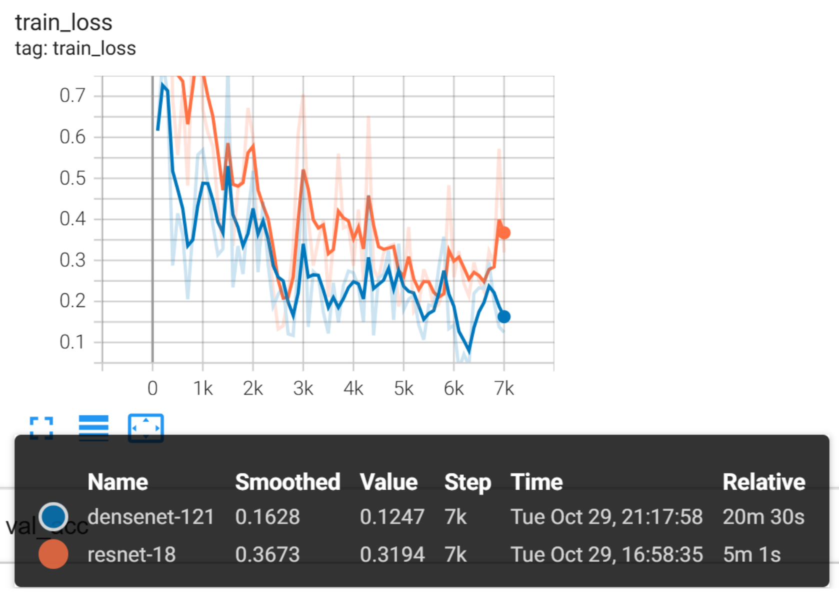
图 1: Conv features

对于 ResNet-18，在前向传播的过程中特征图的尺寸逐渐减小，通道数逐渐增加
在前几层的特征图中，可以看到一些简单的特征，如边缘、颜色等，随着网络深度的增加，特征图变得更加抽象

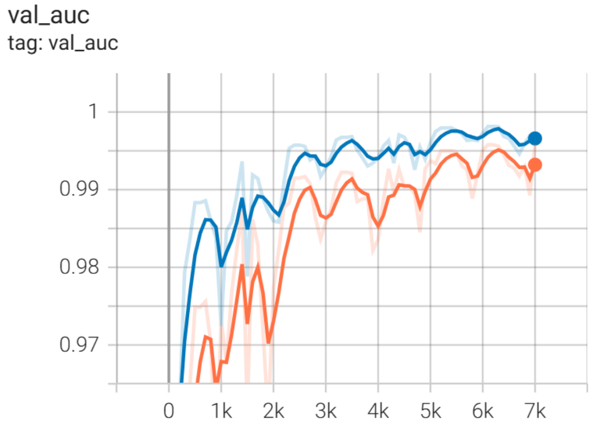
3.3 Train new model

选择 DenseNet 的变体 DenseNet121 在该数据集上从零开始训练, 其可以通过密集连接来增强特征的重用性。训练结果如下:

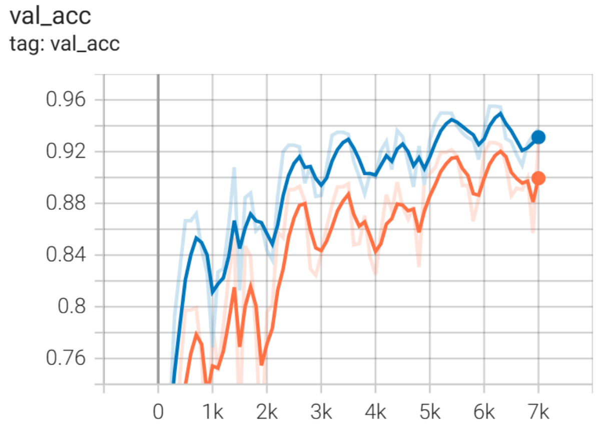
1. train loss



2. valid auc score



3. valid acc score



其中蓝色曲线为 DenseNet121, 红色曲线为 ResNet-18, DenseNet121 的测试结果 auc:0.985,acc:0.864

3.4 Training strategies

使用数据增强和学习率调度策略来提高模型 B 的性能, 不同策略的测试集结果如下:

model	auc	acc
origin	0.985	0.864
StepLR	0.978	0.746
rotation	0.979	0.875
HorizontalFlip	0.983	0.881
rotation+HorizontalFlip	**0.987**	**0.906**

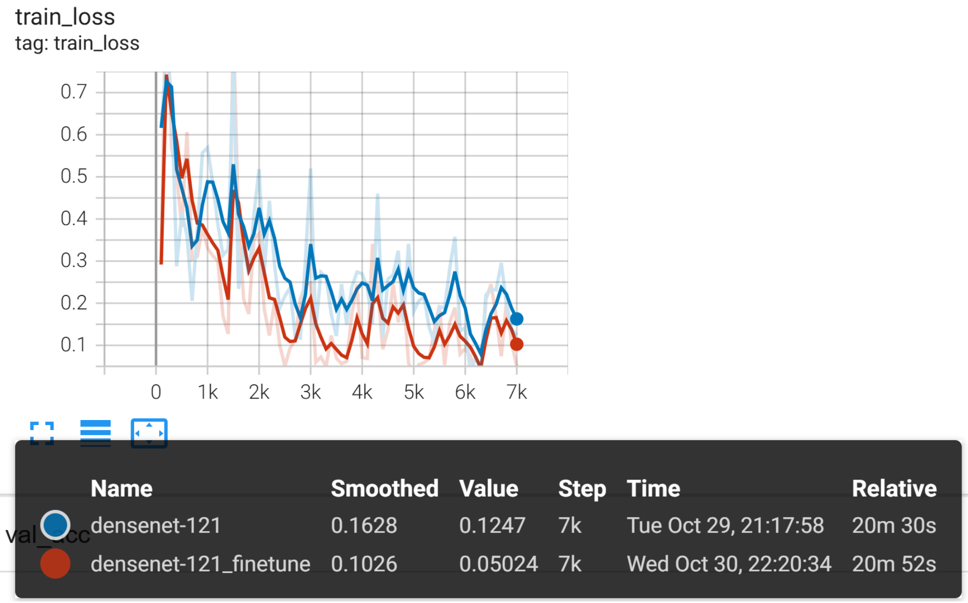
表 3: Training strategies

origin 为未使用数据增强从零开始训练的 DenstNet121, 其学习率调度策略为 ConsineAnnealingLR
其余每行是在 origin 的基础上使用不同的数据增强或学习率调度策略的测试结果
可以看出，对原始数据集使用随机旋转 15 度和水平翻转两种数据增强策略，模型性能有所提升，其中 acc 超过了 0.9

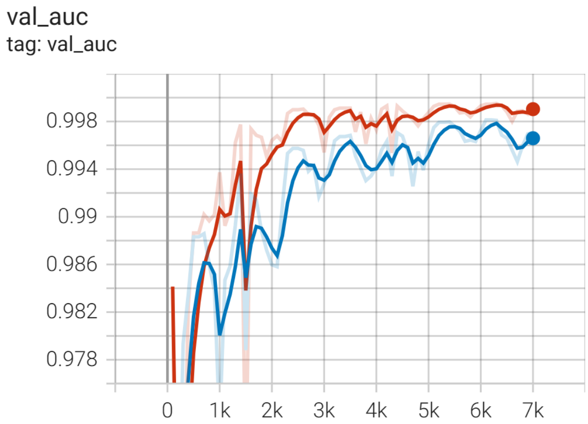
3.5 Fine-tuning a pre-trained model

加载在 ImageNet 上预训练的 DenseNet121 模型权重, 用新的数据集 PathMNIST 进行 fine-tuning, 结果如下:

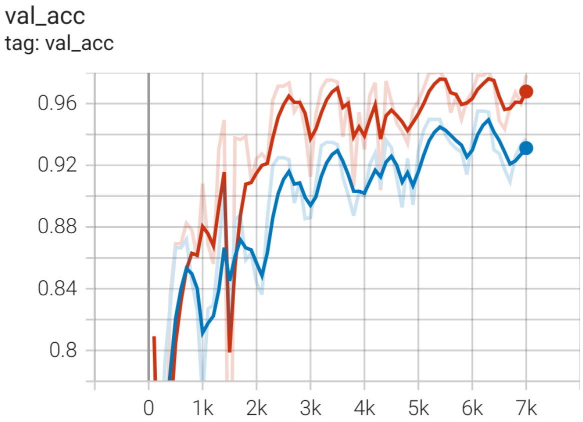
1. train loss



2. valid auc score



3. valid acc score



在测试集上的结果如下：

model	auc	acc
DenseNet121	0.985	0.864
fine-tuning	**0.997**	**0.936**

表 4: Fine-tuning

上图中红色曲线为 fine-tuning, 蓝色曲线为 DenseNet121 从头开始训练的结果, 从训练曲线和测试结果可以看出 DenseNet121 fine-tuning 的性能优于从头开始训练

fine-tuning 和 train from scratch 的区别还在于：

- fine-tuning 在微调时需要设置较小的学习率，以免破坏预训练模型的参数
- fine-tuning 的收敛速度更快：由于网络已经有了良好的初始化 (预训练权重)，微调时通常能够更快地收敛