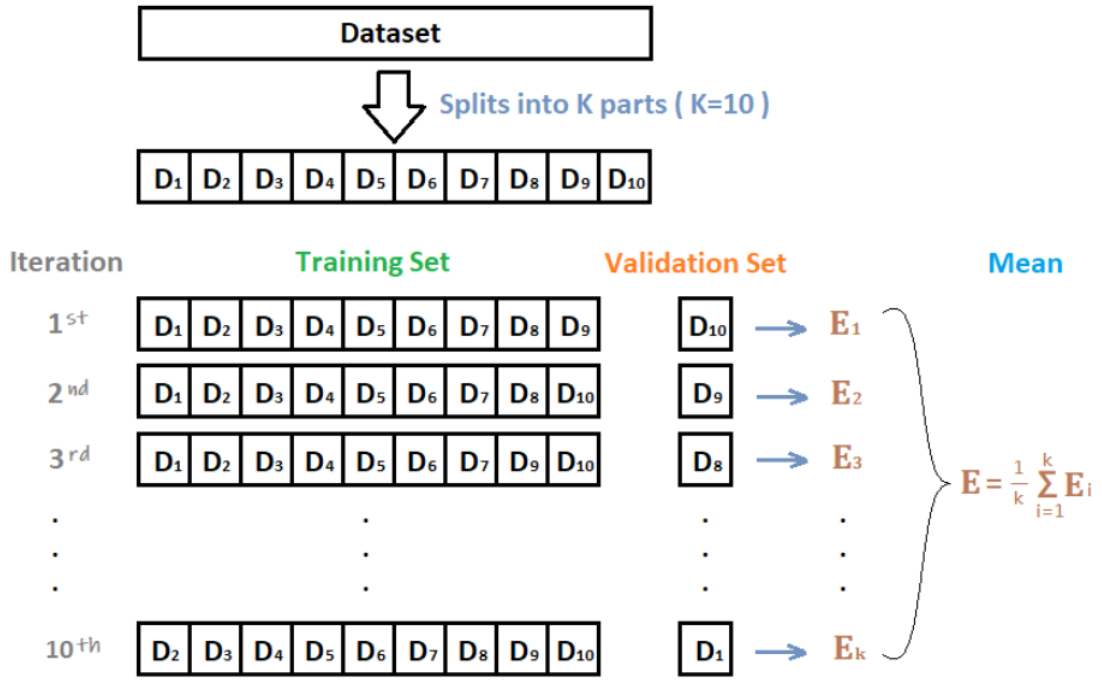
Task5 模型集成

一、集成学习方法

在机器学习中的集成学习可以在一定程度上提高预测精度，常见的集成学习方法有Stacking、Bagging和Boosting，同时这些集成学习方法与具体验证集划分联系紧密。

由于深度学习模型一般需要较长的训练周期，如果硬件设备不允许建议选取留出法，如果需要追求精度可以使用交叉验证的方法。

下面假设构建了10折交叉验证，训练得到10个CNN模型。



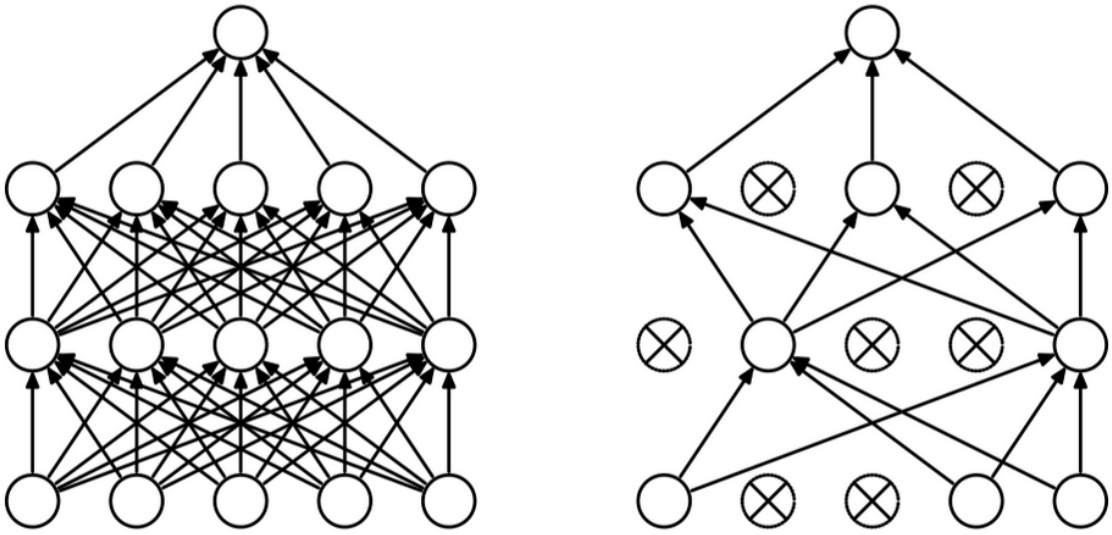
那么在10个CNN模型可以使用如下方式进行集成：

- 对预测的结果的概率值进行平均，然后解码为具体字符；
- 对预测的字符进行投票，得到最终字符。

二、深度学习中的集成学习

1、集成学习

Dropout可以作为训练深度神经网络的一种技巧。在每个训练批次中，通过随机让一部分的节点停止工作。同时在预测的过程中让所有的节点都其作用。



(a) Standard Neural Net　　(b) After applying dropout.

Dropout经常出现在在先有的CNN网络中，可以有效的缓解模型过拟合的情况，也可以在预测时增加模型的精度。

加入Dropout后的网络结构如下：

```python
# 定义模型
class SVHN_Model1(nn.Module):
    def __init__(self):
        super(SVHN_Model1, self).__init__()
        # CNN提取特征模块
        self.cnn = nn.Sequential(
            nn.Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2)),
            nn.ReLU(),
            nn.Dropout(0.25),
            nn.MaxPool2d(2),
            nn.Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2)),
            nn.ReLU(),
            nn.Dropout(0.25),
            nn.MaxPool2d(2),
        )
        #
        self.fc1 = nn.Linear(32*3*7, 11)
        self.fc2 = nn.Linear(32*3*7, 11)
        self.fc3 = nn.Linear(32*3*7, 11)
        self.fc4 = nn.Linear(32*3*7, 11)
        self.fc5 = nn.Linear(32*3*7, 11)
        self.fc6 = nn.Linear(32*3*7, 11)

    def forward(self, img):
        feat = self.cnn(img)
        feat = feat.view(feat.shape[0], -1)
        c1 = self.fc1(feat)
        c2 = self.fc2(feat)
        c3 = self.fc3(feat)
        c4 = self.fc4(feat)
        c5 = self.fc5(feat)
        c6 = self.fc6(feat)
        return c1, c2, c3, c4, c5, c6
```

2、TTA

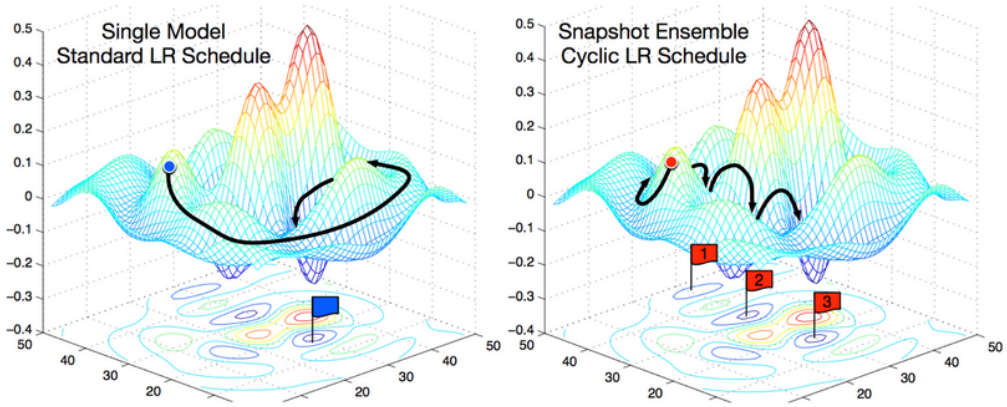
测试集数据扩增（Test Time Augmentation，简称TTA）也是常用的集成学习技巧，数据扩增不仅可以在训练时候用，而且可以同样在预测时候进行数据扩增，对同一个样本预测三次，然后对三次结果进行平均。

```
def predict(test_loader, model, tta=10):
    model.eval()
    test_pred_tta = None
    # TTA 次数
    for _ in range(tta):
        test_pred = []

        with torch.no_grad():
            for i, (input, target) in enumerate(test_loader):
                c0, c1, c2, c3, c4, c5 = model(data[0])
                output = np.concatenate([c0.data.numpy(), c1.data.numpy(),
                    c2.data.numpy(), c3.data.numpy(),
                    c4.data.numpy(), c5.data.numpy()], axis=1)
                test_pred.append(output)

        test_pred = np.vstack(test_pred)
        if test_pred_tta is None:
            test_pred_tta = test_pred
        else:
            test_pred_tta += test_pred

    return test_pred_tta
```

3、Snapshot

假设我们训练了10个CNN则可以将多个模型的预测结果进行平均。但是加入只训练了一个CNN模型，如何做模型集成呢?

在论文Snapshot Ensembles中，作者提出使用cyclical learning rate进行训练模型，并保存精度比较好的一些checkopint，最后将多个checkpoint进行模型集成。



由于在cyclical learning rate中学习率的变化有周期性变大和减少的行为，因此CNN模型很有可能在跳出局部最优进入另一个局部最优。在Snapshot论文中作者通过使用表明，此种方法可以在一定程度上提高模型精度，但需要更长的训练时间。

	Method	C10	C100	SVHN
ResNet-110	Single model	5.52	28.02	1.96
	NoCycle Snapshot Ensemble	5.49	26.97	1.78
	SingleCycle Ensembles	6.66	24.54	1.74
	Snapshot Ensemble ($\alpha_0 = 0.1$)	5.73	25.55	**1.63**
	Snapshot Ensemble ($\alpha_0 = 0.2$)	**5.32**	**24.19**	1.66
Wide-ResNet-32	Single model	5.43	23.55	1.90
	Dropout	4.68	22.82	1.81
	NoCycle Snapshot Ensemble	5.18	22.81	1.81
	SingleCycle Ensembles	5.95	21.38	1.65
	Snapshot Ensemble ($\alpha_0 = 0.1$)	**4.41**	**21.26**	1.64
	Snapshot Ensemble ($\alpha_0 = 0.2$)	4.73	21.56	**1.51**

三、结果后处理

在不同的任务中可能会有不同的解决方案，不同思路的模型不仅可以互相借鉴，同时也可以修正最终的预测结果。

在本次赛题中，可以从以下几个思路对预测结果进行后处理：

- 统计图片中每个位置字符出现的频率，使用规则修正结果；
- 单独训练一个字符长度预测模型，用来预测图片中字符个数，并修正结果。

集成学习只能在一定程度上提高精度，并需要耗费较大的训练时间，因此建议先使用提高单个模型的精度，再考虑集成学习过程；具体的集成学习方法需要与验证集划分方法结合，Dropout和TTA在所有场景有可以起作用。