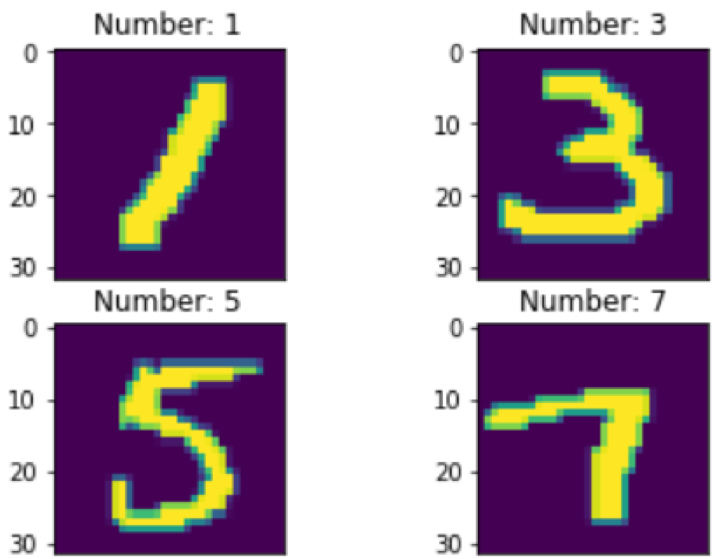
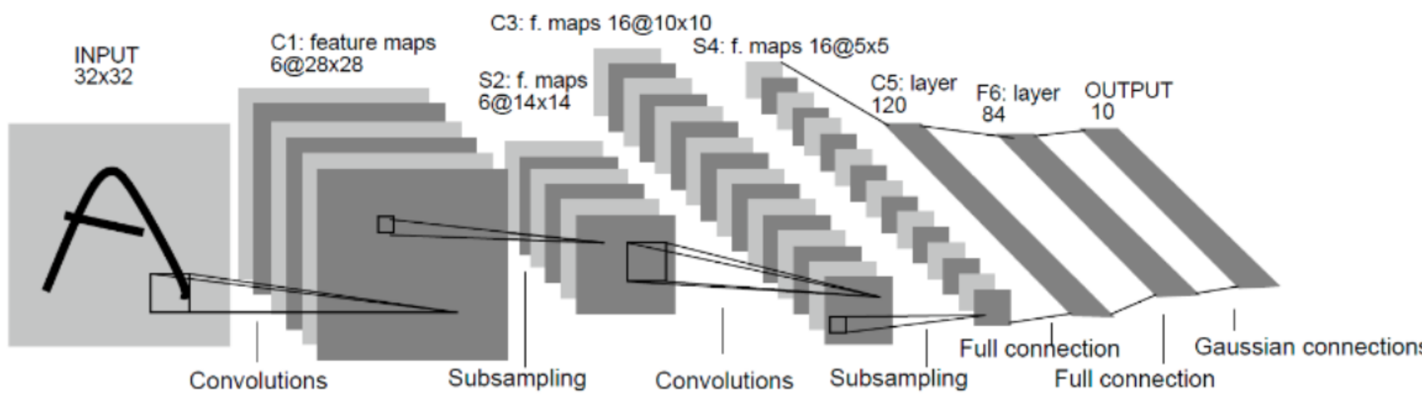
手写数字符兼垃圾分类

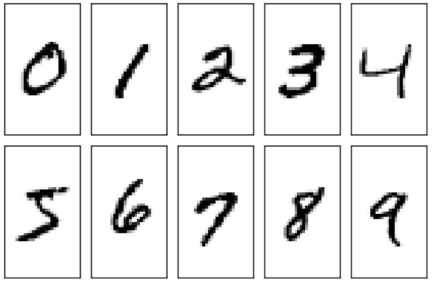
人工智能基础 —— 实践课（五）



- LeNet5 + MNIST被誉为深度学习领域的"Hello world"。本实验主要介绍使用MindSpore在MNIST手写数字数据集上开发和训练一个LeNet5模型，并验证模型精度。
- 根据学习的知识自主设计垃圾分类模型，作业评分测试都将基于垃圾分类进行。
- 附加题: 手机端部署垃圾分类模型并测试



MNIST数据集



MNIST 数据集来自美国国家标准与技术研究所，训练集 (training set) 由来自 250 个不同人手写的数字构成，其中 50% 是高中学生，50% 来自人口普查局 (the Census Bureau) 的工作人员．测试集(test set) 也是同样比例的手写数字数据.

- Training set images: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)
- Training set labels: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)
- Test set images: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)
- Test set labels: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

数据预处理:

1. mini-batch vs. full-batch: 设定批训练大小batch-size
2. Resize: 统一缩放图片到统一标准大小
3. Rescale: 数据归一化
4. HWC2CHW: 将图像矩阵的高宽通道三个维度进行调整对换
5. Others: 旋转, 翻转, 裁剪, 高斯噪声, 灰度转换, 亮度, 饱和度…

mindspore.dataset()
mindspore.dataset.transforms.c_transforms()
mindspore.dataset.vision.c_transforms()

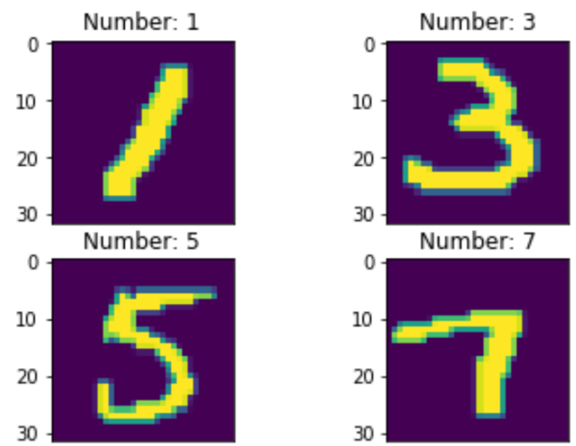
MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

数据预处理

```python
def create_dataset(data_dir, training=True, batch_size=32, resize=(32, 32),
                   rescale=1/(255*0.3081), shift=-0.1307/0.3081, buffer_size=64):
    data_train = os.path.join(data_dir, 'train') # 训练集信息
    data_test = os.path.join(data_dir, 'test') # 测试集信息
    ds = ms.dataset.MnistDataset(data_train if training else data_test)

    ds = ds.map(input_columns=["image"], operations=[CV.Resize(resize), CV.Rescale(rescale, shift), CV.HWC2CHW()])
    ds = ds.map(input_columns=["label"], operations=C.TypeCast(ms.int32))
    # When `dataset_sink_mode=True` on Ascend, append `ds = ds.repeat(num_epochs) to the end
    ds = ds.shuffle(buffer_size=buffer_size).batch(batch_size, drop_remainder=True)

    return ds
```

```python
import matplotlib.pyplot as plt
ds = create_dataset('MNIST', training=False)
data = ds.create_dict_iterator().get_next()
images = data['image'].asnumpy()
labels = data['label'].asnumpy()

for i in range(1, 5):
    plt.subplot(2, 2, i)
    plt.imshow(images[i][0])
    plt.title('Number: %s' % labels[i])
    plt.xticks([])
plt.show()
```

```
(32, 1, 32, 32)
(32,)
```



MindSpore官方文档：https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

数据预处理

mindspore.dataset.MnistDataset(dataset_dir, usage=None, num_samples=None, num_parallel_workers=None, shuffle=None, sampler=None, num_shards=None, shard_id=None, cache=None):

- dataset_dir (str) – 包含数据集的根目录的路径.
- shuffle (bool, optional) – 是否打乱数据集, 默认为不打乱数据集顺序.

mindspore.dataset.MnistDataset.map(input_columns, operations):

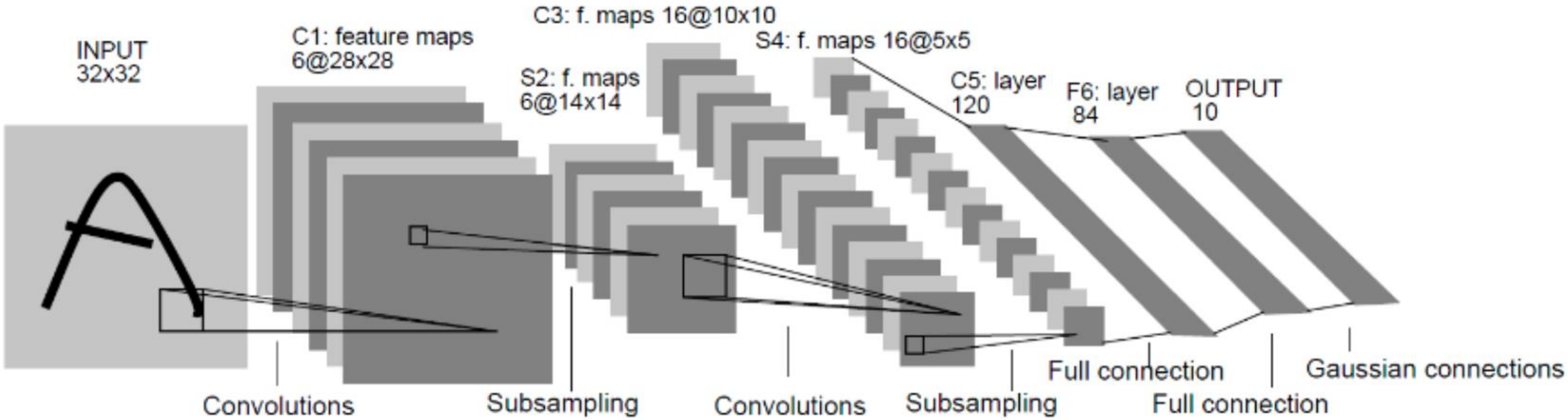
- input_columns=[‘key’]: 对[“image”]或[“label”]执行数据变换
- operations: 数据变换，如 CV.Resize(resize), CV.Rescale(rescale, shift), CV.HWC2CHW(), C.TypeCast(ms.int32)
  其中:
  import mindspore.dataset.transforms.c_transforms as C
  import mindspore.dataset.vision.c_transforms as CV

mindspore.dataset.MnistDataset.shuffle(buffer_size=buffer_size).batch(batch_size, drop_remainder=True)

- buffer_size:将被加入缓冲器的元素的最大数
- batch_size: 一批输入网络训练的样本数量
- drop_remainder:当epoch剩余样本数不足一个batch的时候，直接丢弃还是保留直接作为一个batch

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

LeNet5



mindspore.nn.Conv2d(in_channels, out_channels, kernel_size, stride = 1, pad_mode ='same', padding = 0, dilation = 1, group = 1, has_bias = False, weight_init ='normal', bias_init ='zeros', data_format ='NCHW')

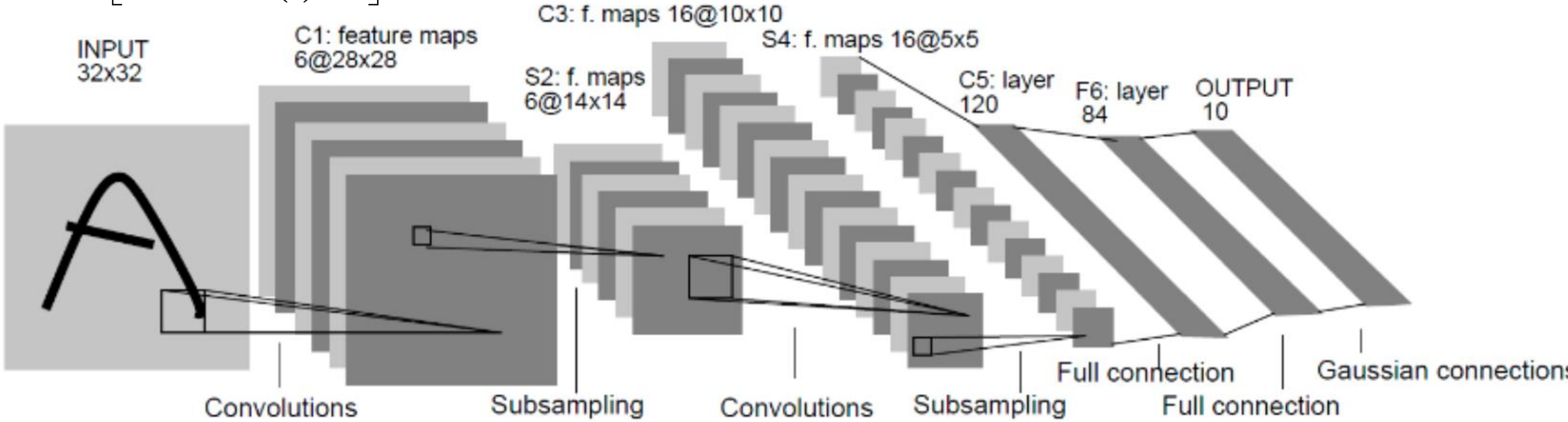
Output Height: $\left\lfloor 1+\frac{H_{in}+2\times\text{padding}-\text{ks\_h}-(\text{ks\_h}-1)\times(\text{dilation-1})}{\text{stride}} \right\rfloor$

Output Weight: $\left\lfloor 1+\frac{H_{in}+2\times\text{padding}-\text{ks\_w}-(\text{ks\_w}-1)\times(\text{dilation-1})}{\text{stride}} \right\rfloor$

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

LeNet5

$$\left[1+\frac{H_{in}(32)-\text{ks\_h}(5)}{\text{stride}(1)}\right]=28 \qquad \left[1+\frac{H_{in}(14)-\text{ks\_h}(5)}{\text{stride}(1)}\right]=10$$



mindspore.nn.Conv2d(in_channels， out_channels， kernel_size， stride = 1， pad_mode ='same'， padding = 0， dilation = 1， group = 1， has_bias = False， weight_init ='normal'， bias_init ='zeros'， data_format ='NCHW')

Output Height: $\left[1+\frac{H_{in}+2\times\text{padding}-\text{ks\_h}-(\text{ks\_h}-1)\times(\text{dilation-1})}{\text{stride}}\right]$

Output Weight: $\left[1+\frac{H_{in}+2\times\text{padding}-\text{ks\_w}-(\text{ks\_w}-1)\times(\text{dilation-1})}{\text{stride}}\right]$

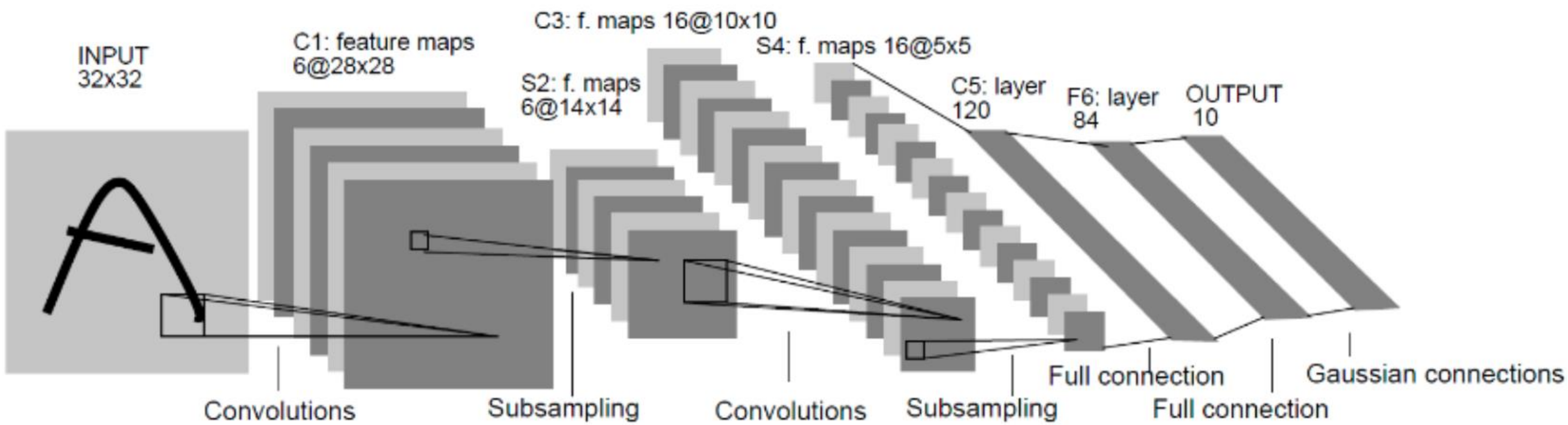
MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

LeNet5

```
class LeNet5(nn.Cell):
    def __init__(self):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, stride=1, pad_mode='valid')
        self.conv2 = nn.Conv2d(6, 16, 5, stride=1, pad_mode='valid')
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = nn.Flatten()
        self.fc1 = nn.Dense(400, 120)
        self.fc2 = nn.Dense(120, 84)
        self.fc3 = nn.Dense(84, 10)

    def construct(self, x):
        x = self.relu(self.conv1(x))
        x = self.pool(x)
        x = self.relu(self.conv2(x))
        x = self.pool(x)
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)

        return x
```

$$\left\lfloor 1+\frac{H_{in}(32)-\text{ks\_h}(5)}{\text{stride}(1)}\right\rfloor = 28$$

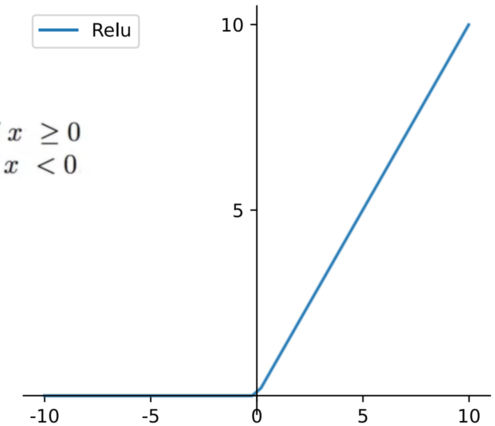
$$\left\lfloor 1+\frac{H_{in}(14)-\text{ks\_h}(5)}{\text{stride}(1)}\right\rfloor = 10$$

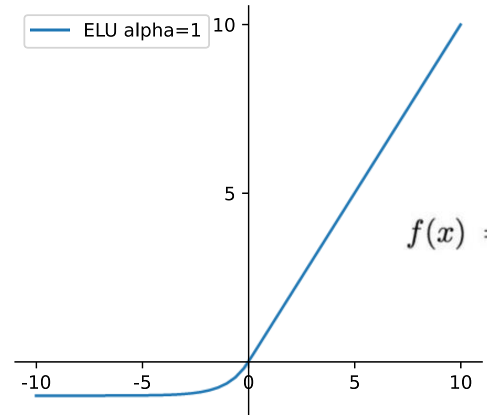


MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

activation

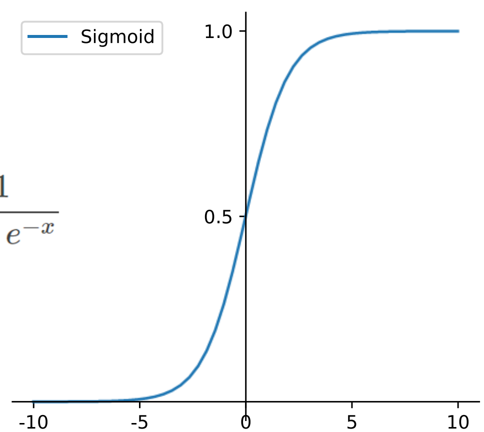
$$y = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

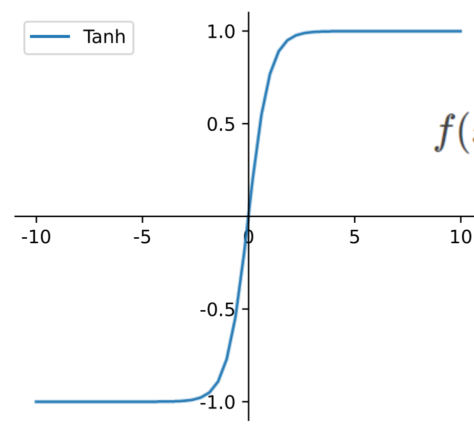




$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}$$

$$f(x) = \frac{1}{1 + e^{-x}}$$





$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

optimal

```python
def train(data_dir, lr=0.01, momentum=0.9, num_epochs=3, ckpt_name="lenet"):
    ds_train = create_dataset(data_dir)
    ds_eval = create_dataset(data_dir, training=False)
    steps_per_epoch = ds_train.get_dataset_size()

    net = LeNet5()
    loss = nn.loss.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
    opt = nn.Momentum(net.trainable_params(), lr, momentum)

    ckpt_cfg = CheckpointConfig(save_checkpoint_steps=steps_per_epoch, keep_checkpoint_max=5)
    ckpt_cb = ModelCheckpoint(prefix=ckpt_name, directory=CKPT_DIR, config=ckpt_cfg)
    loss_cb = LossMonitor(steps_per_epoch)

    model = Model(net, loss, opt, metrics={'acc', 'loss'})
    model.train(num_epochs, ds_train, callbacks=[ckpt_cb, loss_cb], dataset_sink_mode=False)
    metrics = model.eval(ds_eval, dataset_sink_mode=False)
    print('Metrics:', metrics)

train(DATA_PATH)
```

- mindspore.nn.loss.SoftmaxCrossEntropyWithLogits

For each instance N_i, the loss is given as:

$$\ell(x_i, t_i) = -\log\left(\frac{\exp(x_{t_i})}{\sum_j \exp(x_j)}\right) = -x_{t_i} + \log\left(\sum_j \exp(x_j)\right),$$

where x_i is a 1D score Tensor, t_i is a scalar.

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

optimal

```python
def train(data_dir, lr=0.01, momentum=0.9, num_epochs=3, ckpt_name="lenet"):
    ds_train = create_dataset(data_dir)
    ds_eval = create_dataset(data_dir, training=False)
    steps_per_epoch = ds_train.get_dataset_size()

    net = LeNet5()
    loss = nn.loss.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
    opt = nn.Momentum(net.trainable_params(), lr, momentum)

    ckpt_cfg = CheckpointConfig(save_checkpoint_steps=steps_per_epoch, keep_checkpoint_max=5)
    ckpt_cb = ModelCheckpoint(prefix=ckpt_name, directory=CKPT_DIR, config=ckpt_cfg)
    loss_cb = LossMonitor(steps_per_epoch)

    model = Model(net, loss, opt, metrics={'acc', 'loss'})
    model.train(num_epochs, ds_train, callbacks=[ckpt_cb, loss_cb], dataset_sink_mode=False)
    metrics = model.eval(ds_eval, dataset_sink_mode=False)
    print('Metrics:', metrics)

train(DATA_PATH)
```

- mindspore.nn.Momentum $v_t = v_{t-1} * u + gradients$

If use_nesterov is True:

$$p_t = p_{t-1} - (grad * lr + v_t * u * lr)$$

If use_nesterov is Flase:

$$p_t = p_{t-1} - lr * v_t$$

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

模型参数保存

```python
def train(data_dir, lr=0.01, momentum=0.9, num_epochs=3, ckpt_name="lenet"):
    ds_train = create_dataset(data_dir)
    ds_eval = create_dataset(data_dir, training=False)
    steps_per_epoch = ds_train.get_dataset_size()

    net = LeNet5()
    loss = nn.loss.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
    opt = nn.Momentum(net.trainable_params(), lr, momentum)

    ckpt_cfg = CheckpointConfig(save_checkpoint_steps=steps_per_epoch, keep_checkpoint_max=5)
    ckpt_cb = ModelCheckpoint(prefix=ckpt_name, directory=CKPT_DIR, config=ckpt_cfg)
    loss_cb = LossMonitor(steps_per_epoch)

    model = Model(net, loss, opt, metrics={'acc', 'loss'})
    model.train(num_epochs, ds_train, callbacks=[ckpt_cb, loss_cb], dataset_sink_mode=False)
    metrics = model.eval(ds_eval, dataset_sink_mode=False)
    print('Metrics:', metrics)

train(DATA_PATH)
```

ModelCheckpoint可以保存模型参数，以便进行再训练或推理。 LossMonitor可以在日志中输出loss，方便用户查看，同时它还会监控训练过程中的loss值变化情况，当loss值为Nan或Inf时终止训练。

MindSpore官方文档: https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

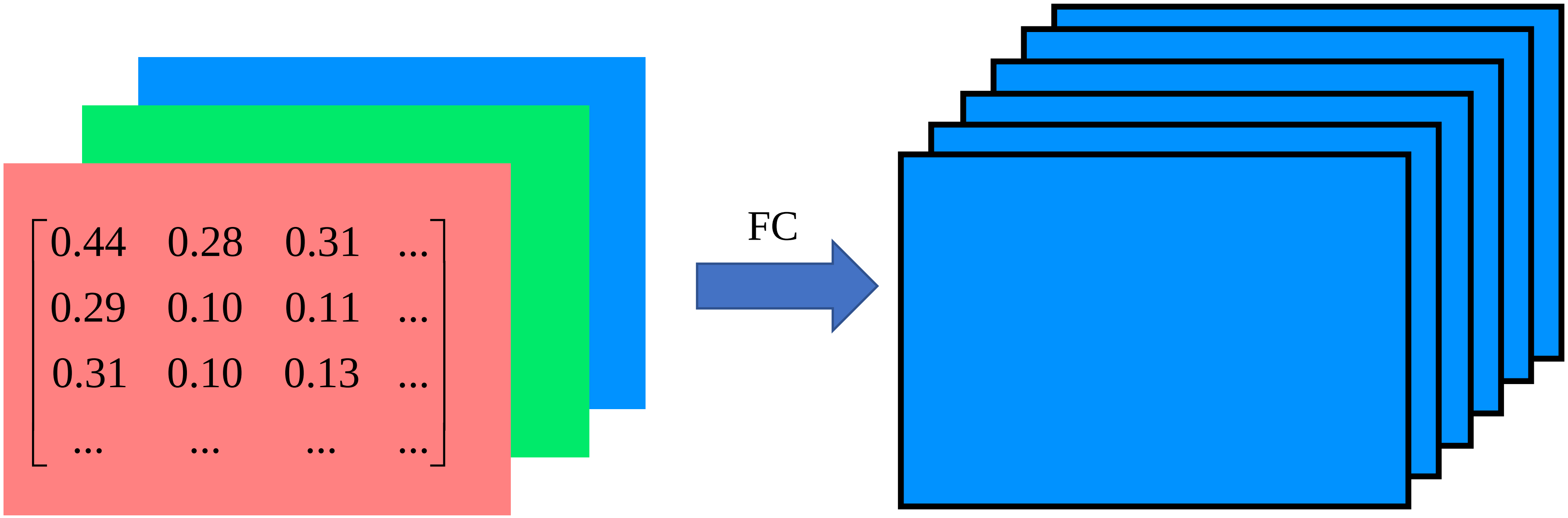
加载Checkpoint进行验证

```
CKPT = os.path.join(CKPT_DIR, 'lenet-3_1875.ckpt')

def eval(data_dir):
    ds_eval = create_dataset(data_dir, training=False)
    net = LeNet5()
    loss = nn.loss.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
    load_checkpoint(CKPT, net=net)
    model = Model(net, loss, metrics={'acc', 'loss'})
    metric = model.eval(ds_eval, dataset_sink_mode=False)
    print(metric)

eval(DATA_PATH)
```

```
def infer(data_dir):
    ds = create_dataset(data_dir, training=False).create_dict_iterator(output_numpy=True)
    data = ds.get_next()
    images = data['image']
    labels = data['label']
    net = LeNet5()
    load_checkpoint(CKPT, net=net)
    model = Model(net)
    output = model.predict(Tensor(data['image']))
    preds = np.argmax(output.asnumpy(), axis=1)

    for i in range(1, 5):
        plt.subplot(2, 2, i)
        plt.imshow(np.squeeze(images[i]))
        color = 'blue' if preds[i] == labels[i] else 'red'
        plt.title("prediction: {}, truth: {}".format(preds[i], labels[i]), color=color)
        plt.xticks([])
    plt.show()

infer('MNIST')
```

MindSpore官方文档：https://www.mindspore.cn/doc/api_python/zh-CN/master/index.html

全连接神经网络的局限性

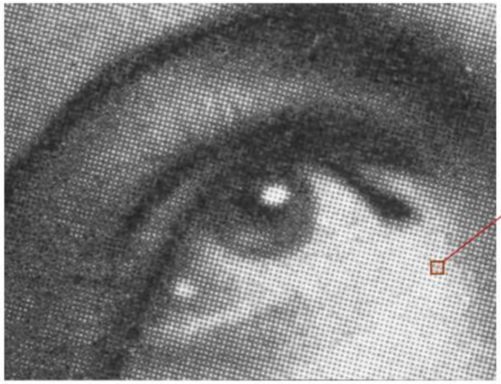


如果只有两个隐藏层，每层256个节点，则MNIST数据集所需要的参数是（28*28*256+256*256+256*10）个w，再加上（256+256+10）个b。

- 如果输入图像大小宽高为1000像素，仅一层全连接就需要1000*1000*256个参数，约等于2亿个w。
- 而卷积神经网络使用了参数共享的方式，换一个角度来解决问题，不仅在准确率上大大提升，也把参数降下来。

《人工智能导论》P214 ~ P223

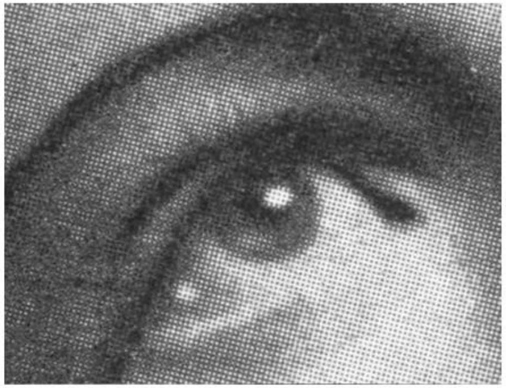
卷积核



这些噪点，属于高频信号

高频信号，就好像平地耸立的山峰

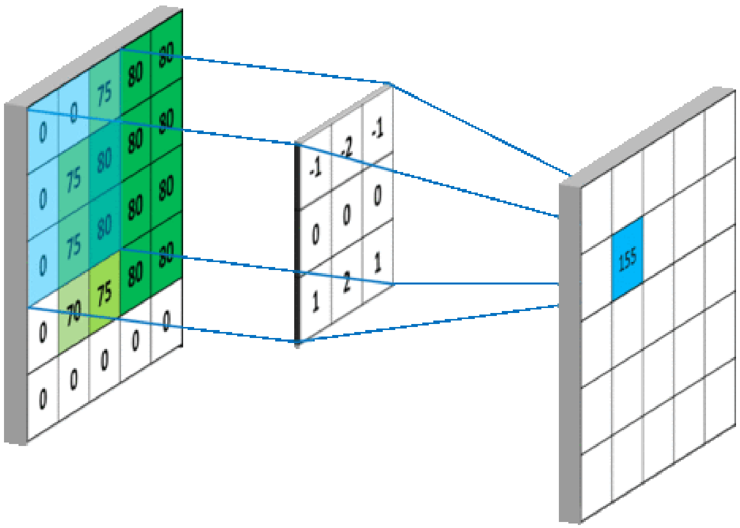
平滑这座山峰的办法之一就是，把山峰刨掉一些土，填到山峰周围去。用数学的话来说，就是把山峰周围的高度平均一下。即把高频信号与周围的数值平均一下就可以平滑图像。



$$\Longrightarrow \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,0} & a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

$$g = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

卷积核



a, b的下标相加都为$1, 1$

$$f = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & \boxed{a_{2,2}} \end{bmatrix} \qquad g = \begin{bmatrix} \boxed{b_{-1,-1}} & b_{-1,0} & b_{-1,1} \\ b_{0,-1} & b_{0,0} & b_{0,1} \\ b_{1,-1} & b_{1,0} & b_{1,1} \end{bmatrix}$$

$$\begin{aligned} c_{1,1} = {} & a_{0,0}b_{1,1} + a_{0,1}b_{1,0} + a_{0,2}b_{1,-1} + a_{1,0}b_{0,1} \\ & + a_{1,1}b_{0,0} + a_{1,2}b_{0,-1} + a_{2,0}b_{-1,1} \\ & + a_{2,1}b_{-1,0} + a_{2,2}b_{-1,-1} \end{aligned}$$

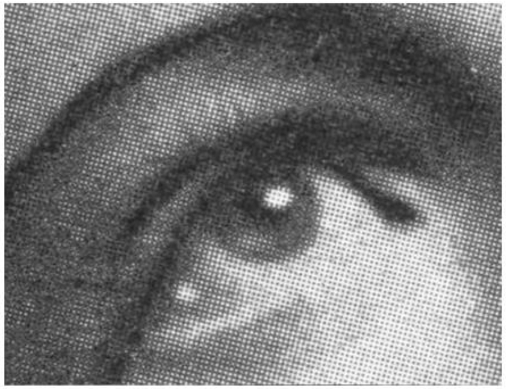
卷积公式形式:

$$(f * g)(1, 1) = \sum_{k=0}^{2} \sum_{h=0}^{2} f(h, k) g(1-h, 1-k)$$

加权求和-卷积滤波

离散卷积本质就是一种加权求和.

卷积神经网络中的卷积操作本质上就是利用一个共享参数的卷积核(*filter*)，**通过逐点计算中心像素点以及相邻像素点的加权和来构成feature map实现空间特征的提取**。



图像数值矩阵

$$\Longrightarrow \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,0} & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,0} & a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

卷积核

$$g = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

MobileNetV2 垃圾分类

本实验以MobileNetV2+垃圾分类数据集为例，主要介绍如在使用 MindSpore在CPU/GPU 平台上进行 Fine-Tune。

垃圾分类信息：

{ '干垃圾': ['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服'],
'可回收物': ['报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张'],
'湿垃圾': ['菜叶', '橙皮', '蛋壳', '香蕉皮'],
'有害垃圾': ['电池', '药片胶囊', '荧光灯', '油漆桶’] }

['贝壳', '打火机', '旧镜子', '扫把', '陶瓷碗', '牙刷', '一次性筷子', '脏污衣服',
'报纸', '玻璃制品', '篮球', '塑料瓶', '硬纸板', '玻璃瓶', '金属制品', '帽子', '易拉罐', '纸张',
'菜叶', '橙皮', '蛋壳', '香蕉皮',
'电池', '药片胶囊', '荧光灯', '油漆桶’]

['Seashell', 'Lighter','Old Mirror', 'Broom','Ceramic Bowl', 'Toothbrush','Disposable Chopsticks','Dirty Cloth’, 'Newspaper', 'Glassware', 'Basketball', 'Plastic Bottle', 'Cardboard','Glass Bottle', 'Metalware', 'Hats', 'Cans', 'Paper’, 'Vegetable Leaf','Orange Peel', 'Eggshell','Banana Peel’, 'Battery', 'Tablet capsules','Fluorescent lamp', 'Paint bucket']

附加题 – 手机端部署模型

MindSpore提供了面想手机及IoT设备的高性能、轻量化端侧推理框架。访问MindSpore Lite官网了解更多：https://www.mindspore.cn/lite/

通过本实验可以了解如何在端侧利用MindSpore Lite C++ API（Android JNI）以及MindSpore Lite图像分类模型完成端侧推理，实现对设备摄像头捕获的内容进行分类，并在App图像预览界面中显示出最可能的分类结果。

实验教程和代码示例请下载：mobilenetv2_android.zip[https://share-course.obs.cn-north-4.myhuaweicloud.com/materials/mobilenetv2_android.zip]。另外，也可以参考MindSpore官网教程实现一个图像分类应用[https://www.mindspore.cn/tutorial/lite/zh-CN/r1.0/quick_start/quick_start.html]。

姓名_学号_手机端模型部署情况.rar /姓名_学号_手机端模型部署情况.zip