به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام و نام خانوادگی	محمد ناصری – مریم عباس زاده
شماره دانشجویی	810100406 - 810100486
تاریخ ارسال گزارش	۱۴۰۱.۰۹.۰۲

فهرست

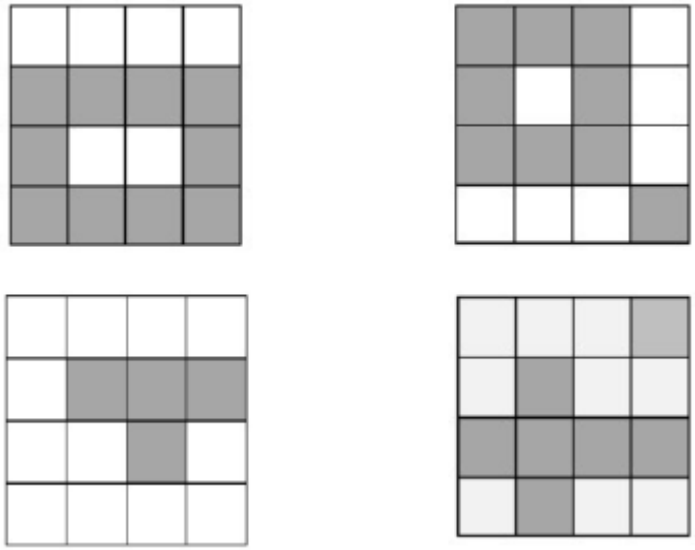
شکل‌ها

جدول‌ها

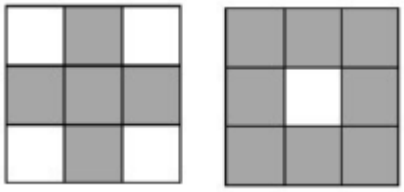
پاسخ ۱. عنوان پرسش اول به فارسی

۱-۱. دست گرمی

برای آشنایی اولیه با CNN ها بصورت تئوری ابتدا یک مسئله تحلیلی را حل می کنیم و در ادامه به خود مقاله می پردازیم : فرض کنید برای مسئله طبقه بندی زیر 4 تصویر 4*4 متعلق به دو کلاس مشخص هستند (دو تصویر ردیف اول کلاس 1 و دو عکس ردیف دوم کلاس ۲) همچنین فرض بفرمایید هر خانه سیاه عدد "1 " و هر خانه سفید عدد " 0 " باشد.



فیلتر ۳*۳ زیر را نیز در نظر میگیریم:



با استفاده از دو فیلتر داده شده و با فرض اینکه عدد بایاس فیلترها -2 باشد و از تابع رلو استفاده کنیم و نهایتا از یک maxpooling با ابعاد ۲ * ۲ استفاده کنیم خروجی نگاشت های ویژگی با Stride 1 و خروجی لایه maxpooling هر یک از تصاویر را بدست آورید.

داده‌ها و فیلترها را به صورت زیر تعریف میکنیم

```
input1 = np.array([[0,0,0,0],
          [1,1,1,1],
          [1,0,0,1],
          [1,1,1,1]])
input2 = np.array([[1,1,1,0],
          [1,0,1,0],
          [1,1,1,0],
          [0,0,0,1]])
input3 = np.array([[0,0,0,0],
          [0,1,1,1],
          [0,0,1,0],
          [0,0,0,0]])
input4 = np.array([[0,0,0,1],
          [0,1,0,0],
          [1,1,1,1],
          [0,1,0,0]])
filter1 = np.array([[0,1,0],
           [1,1,1],
           [0,1,0]])
filter2 = np.array([[1,1,1],
           [1,0,1],
           [1,1,1]])
bias = -2
```

سپس با کمک فانکشن‌های زیر خروجی را برای هر قسمت چاپ میکنیم:

```
output_height = input1.shape[0] - filter.shape[0] + 1
output_width =  input1.shape[1] - filter.shape[1] + 1
def Stride1(inputx, filter):
    output = np.zeros((output_height,output_width))
    for i in range(output_height):
      for j in range(output_width):
        for ii in range(filter.shape[0]):
          for jj in range(filter.shape[1]):
            output[i,j] += inputx[i+ii,j+jj] * filter[ii,jj]
        output[i,j] += bias
    return output


def MaxPooling(inputx):
    output = np.zeros((output_height,output_width))
    for m,i in enumerate(range(0,inputx.shape[0],2)):
      for n,j in enumerate(range(0,inputx.shape[1],2)):
        temp = []
        for ii in range(output_width):
          for jj in range(output_width):
            temp.append(inputx[i+ii,j+jj])
        output[m,n] = max(temp)
    return output
```

خروجی دو حالت به صورت زیر است:

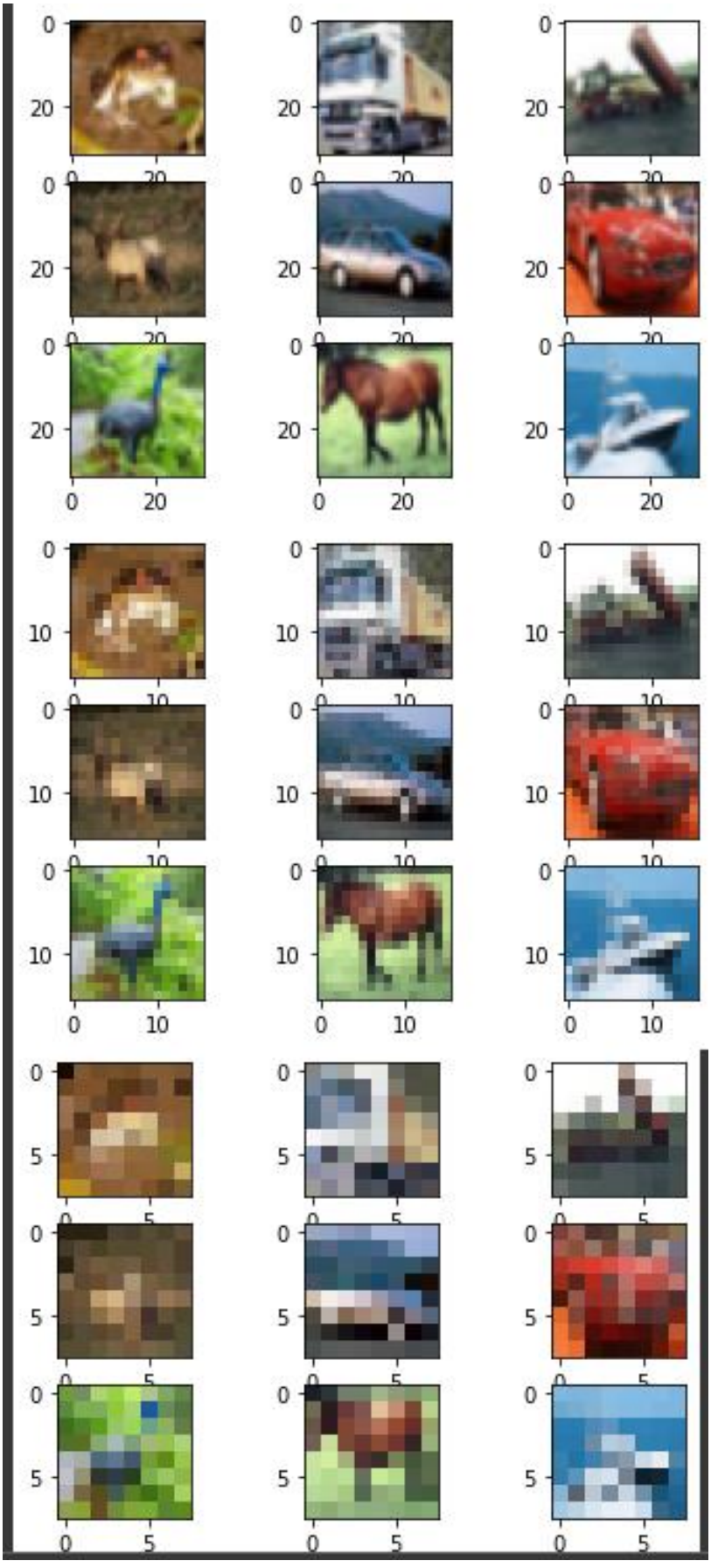
```
MaxPooling of input1 is:
 [[1. 1.]
 [1. 1.]]
MaxPooling of input2 is:
 [[1. 1.]
 [1. 1.]]
MaxPooling of input3 is:
 [[1. 1.]
 [0. 1.]]
MaxPooling of input4 is:
 [[1. 1.]
 [1. 1.]]
```

```
Stride1 of input1, filter1 is:
 [[1. 1.]
 [1. 1.]]
Stride1 of input1, filter2 is:
 [[1. 1.]
 [5. 5.]]
Stride1 of input2, filter1 is:
 [[2. 1.]
 [1. 1.]]
Stride1 of input2, filter2 is:
 [[6. 2.]
 [2. 1.]]
Stride1 of input3, filter1 is:
 [[0. 2.]
 [0. 0.]]
Stride1 of input3, filter2 is:
 [[0. 1.]
 [1. 1.]]
Stride1 of input4, filter1 is:
 [[0. 0.]
 [3. 1.]]
Stride1 of input4, filter2 is:
 [[1. 3.]
 [2. 2.]]
```

۲-۱ تاثیر تغییر رزلوشن در طبقه بندی CNN

هدف در این تمرین مقایسه نتیجه طبقه بندی مناسب با استفاده از شبکه CNN برای مجموعه داده‌های CIFAR–10 با رزولوشن های متفاوت است. مجموعه CIFAR-10 شامل 60 هزار تصویر رنگی است که در 10 کلاس دسته بندی شده و ابعاد تصاویر آن ۳۲ * ۳۲ می باشد. ابتدا مقاله مربوط به این سوال را با دقت مطالعه فرمایید که با کلیک بر روی این لینک قابل دانلود است. و سپس به سوالات در ادامه پاسخ دهید.

برای حل این مساله ابتدا پس از فراخوانی دیتاست مورد نظر به کمک کتابخانه OpenCV به تغییر سایز عکس‌ها میپردازیم. نمونه تصاویر ریسایز شده به شکل زیر هستند:



بعد از این مرحله برای انجام TOTV ابتدا مدل را برای ۳۲*۳۲ فیت میکنیم و سپس برای رزولوشن‌های دیگیر ابتدا تصاویر را به رزولوشن مورد نظر برده و سپس برمیگردانیم به رزولوشن۳۲ و تست میکنیم.
همچنین داده‌ها را نرمالایز نیز انجام میدهیم به کمک تابع زیر:

```
# scale pixels
def prep_pixels(train, test):
    # convert from integers to floats
    train_norm = train.astype('float32')
    test_norm = test.astype('float32')
    # normalize to range 0-1
    train_norm = train_norm / 255.0
    test_norm = test_norm / 255.0
    # return normalized images
    return train_norm, test_norm
```

برای قسمت TVTV نیز سه مدل مجزا تعریف و اجرا میکنیم. نتایج کلی به صورت زیر است:

	TOTV			TVTV		
	Accuracy	Precision	F1	Accuracy	Precision	F1
32*32	0.825	0.848	0.828	0.829	0.852	0.832
16*16	0.484	0.522	0.480	0.757	0.811	0.756
8*8	0.1	0.127	0.037	0.616	0.731	0.891

پاسخ ۲ – آشنایی با معماری شبکه CNN

هدف از این تمرین طبقه‌بندی دیتاست Fashion-mnist با استفاده از شبکه CNN است. به این منظور ابتدا مقاله "CNN Model for Image Classification on MNIST and Fashion MNIST Dataset" را که پیوست فایل تکلیف شده است را مطالعه کنید. در این مقاله معماری طبقه بندی دو دیتاست با 5 معماری مختلف CNN آورده شده است. پس از مطالعه مقاله به سوالات زیر پاسخ دهید.

۳-۱. لود دیتاست مقاله

مسئله طبقه بندی لباس Fashion-MNIST یک مجموعه داده استاندارد جدید است که در بینایی کامپیوتر و یادگیری عمیق استفاده می شود.

اگرچه مجموعه داده نسبتا ساده است، اما می تواند به عنوان پایه ای برای یادگیری و تمرین نحوه توسعه، ارزیابی و استفاده از شبکه های عصبی کانولوشنال عمیق برای طبقه بندی تصاویر استفاده شود.

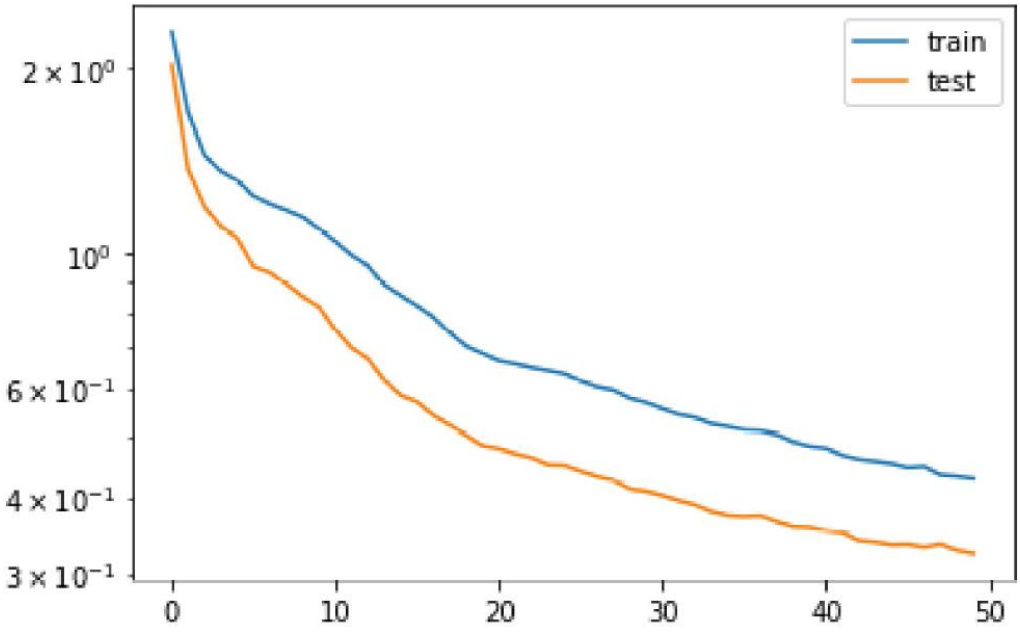
مجموعه داده Fashion-MNIST به عنوان یک مجموعه داده جایگزین چالش برانگیزتر برای مجموعه داده MNIST پیشنهاد شده است. این مجموعه داده ای متشکل از 60000 تصویر مربعی کوچک 28×28 پیکسلی در مقیاس خاکستری از آیتم های 10 نوع لباس، مانند کفش، تی شرت، لباس، و غیره است. نگاشت تمام اعداد صحیح 0-9 به برچسب های کلاس در زیر فهرست شده است.

- 0: تی شرت/تاپ
- 1: شلوار
- 2: پولور
- 3: لباس پوشیدن
- 4: کت
- 5: صندل
- 6: پیراهن
- 7: کفش ورزشی
- 8: کیف
- 9: نیم بوت

برای این حل از معماری‌های ۳ و ۵ استفاده کردیم که عبارتند از:

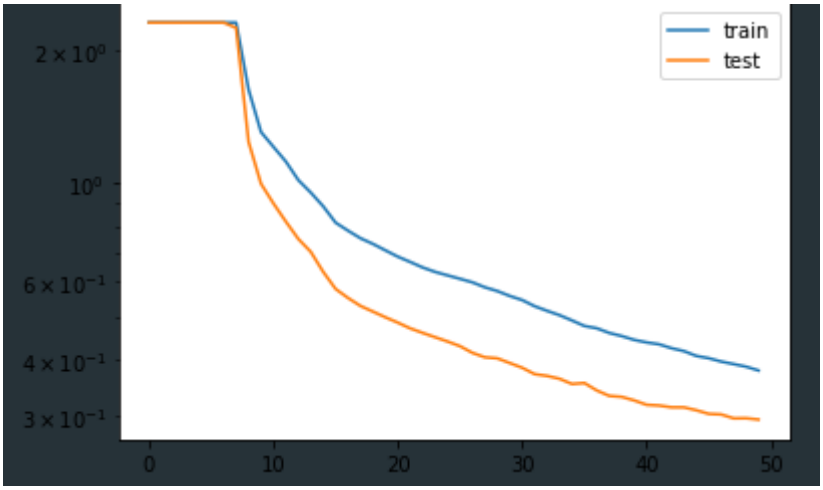
Architecture 5	Architecture 3
4 convolutional layers with (3 x 3) filter size and 2 fully connected layers	3 convolutional layers with (2 x 2) filter size and 2 fully connected layers
(1) INPUT:28×28×1 (2) FC:10 Output Classes	(1) INPUT:28×28×1 (2) FC:10 Output Classes
(3) CONV2D:3×3 size,32 filters (4) CONV2D:3×3 size,32 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:3×3 size,64 filters (7) CONV2D:3×3 size,64 filters (8) POOL:2×2 size (9) DROPOUT: = 0.25 (10) FC:512 Hidden Neurons (11) DROPOUT: = 0.5	(3) CONV2D:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D:2×2 size,64 filters (7) POOL:2×2 size (8) DROPOUT: = 0.25 (9) CONV2D :2×2 size,64 filters (10) DROPOUT: = 0.25 (11) FC:64 Hidden Neurons (12) DROPOUT: = 0.25

نتایج معماری سوم به شرح زیر بود:



Train			Test		
Accuracy	Precision	F1	Accuracy	Precision	F1
0.899	0.10	0.10	0.881	0.10	0.10

همچنین نتایج معماری پنجم به شرح زیر بود:



Train			Test		
Accuracy	Precision	F1	Accuracy	Precision	F1
0.90	0.09	0.09	0.89	0.09	0.09

استفاده از dropout در مدل ما باعث افزایش دقت و کاهش مقدار ضرر شد.

مزیت اصلی روش حذف این است که مانع از همگام سازی تمام نورون های یک لایه می شود که وزن خود را به طور همزمان بهینه کنند. این انطباق، که در گروه های تصادفی انجام می شود، از همه موارد جلوگیری می کند

نورون‌ها به یک هدف همگرا می‌شوند، بنابراین وزن‌ها را به هم مرتبط می‌کنند.

ویژگی دوم کشف شده برای استفاده از حذف این است که فعال سازی های پنهان است

واحدها پراکنده می شوند که این نیز یک ویژگی مطلوب است.