دانشگاه فردوسی مشهد
یادگیری عمیق – نام دانشجو:

مدرس: **محمود امین‌طوسی**
۱۴۰۲/۱۱/۰۴ **زمان: ۱۰۰ دقیقه**

• امتحان شامل ۴ صفحه و ۵ پرسش است. • استفاده از مداد مشکی اشکالی ندارد. • قوانین آزمون رعایت شوند.

پرسش	۱	۲	۳	۴	۵	جمع نمرات
بارم	۲۰	۳۰	۷۰	۴۰	۴۰	۲۰۰
نمره						

۱. (۲۰ نمره) موضوع ارائه هر یک از دوستان شما چه بود؟ جلوی نام هر فرد موضوع ارائه وی را نوشته و اگر در ارائه ایشان تشریف داشته‌اید یک نمره (از ۲۰) بدهید. آقایان و خانمها به ترتیب الفبا:

نام دانشجو	موضوع ارائه	نمره
پیشرو		
جلوگیر		
رعنائی		
سجادی		
علی آبادی		
فاتحی‌نیا		
فکور		
مبشری		
محمدی		
نجیب		
هوسی		

۲. (۳۰ نمره) شکل زیر را توضیح دهید. در کجای برنامه مات کردن و شفاف کردن تصویر از این تکنیک استفاده شد؟ ماتریس را کامل کنید. نتیجه ضرب ماتریس و نتیجه کانولوشن که بردار حاصل، هم‌اندازه با بردار ورودی باشد را بنویسید.

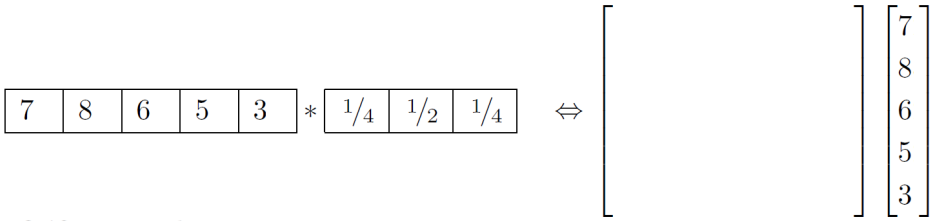

Figure 3.12 *One-dimensional signal convolution as a sparse matrix-vector multiplication*

۳. فرض کنید یک مجموعه داده از تصاویر ارقام صفر تا ۲ که همگی 5×5 پیکسل هستند دارید و دو شبکه‌ی زیر برای طبقه‌بندی این داده‌ها پیشنهاد شده است:

۱. یک شبکه عصبی ساده که لایه اول ورودی ۲۵ تایی و لایه‌ی دوم (آخر) ۳ تایی برای طبقه‌بندی دارد.

۲. یک شبکه‌ی عصبی پیچشی که تصاویر را به همان شکل اولیه گرفته و روی آنها ۳ فیلتر 5×5 اعمال کنیم.

در ادامه تکه برنامه‌هایی برای هر دو مورد بالا نوشته شده است. ابتدا همه‌ی پرسش‌های این سؤال را ملاحظه فرموده و سپس پاسخ دهید.

```
1  len = 5
2  model1 = keras.Sequential()
3  model1.add(keras.Input(shape=(len*len,)))
4  model1.add(layers.Dense(3, activation="softmax"))
5  model1.summary()
6
7  img = np.random.rand(len, len)
8  X = tf.reshape(img, (1, len * len))
9  print(X.shape)
10 y = model1.predict(X)
11 print(y.shape)
12
13 print(y) # Only here suppose that all of the weights of model1
       are zeros and the activation function is sigmoid.
14
15 model2 = keras.Sequential()
16 model2.add(layers.Conv2D(3, (5, 5), activation="softmax",
      input_shape=(len, len, 1)))
17 model2.summary()
18
19 X = tf.reshape(img, (1, len, len, 1))
20 print(X.shape)
21 y = model2.predict(X)
22 print(y.shape)
23 print(y) # Write the range of the outputs
```

(آ) (۲۰ نمره) مشخص کنید کدام بخش برنامه مرتبط با هر یک از دو پیشنهاد فوق‌الذکر هست. برای بخش مرتبط با هر پیشنهاد یک توضیح مختصر بنویسید. آیا کد داده شده، مطابق با پیشنهاد هست؟ (اگر در پاسخنامه پاسخ می‌دهید، فقط شماره خط برنامه ذکر شود، کفایت می‌کند).

(ب) (۳۰ نمره) خروجی دستوراتی که خروجی دارند به همراه شماره خط مربوطه را بنویسید. برای دستورات چاپ ساختار مدل، Output Shape و تعداد پارامترها را بنویسید.

(ج) (۲۰ نمره) آیا دو شبکه از نظر عملکردی با هم متفاوت هستند؟ توضیحی بنویسید.

۴. (۴۰ نمره) بخشی از تمرین کانولوشن دو بعدی به صورت زیر بود:

...در این تمرین مشابه تمرین قبل که برای حالت یک بعدی باید فیلتر اصلی پیدا می‌شد، برای حالت دو بعدی فیلتر را بیابید:

۱. ابتدا سعی کنید یک روش کلاسیک بهینه‌سازی برای حل $\min_x \|Ax - b\|$ بیابید.
۲. به جای ماتشدگی از نوع حرکت، عمل استخراج لبه‌های افقی (یا عمودی) را روی تصاویر اعمال کنید
۳. این سه برنامه را به گونه‌ای ترکیب کنید که یک تصویر توسط کتابخانه سیگنال با کرنل دانسته (مثلا پرویت) فیلتر شود، سپس با یک مدل کراس و با داشتن تصویر اصلی و تصویر حاصل از کانولوشنِ فیلتر روی تصویر (که لبه‌های تصویر هست)، کرنلی را بیابید که همان کار را روی تصویر انجام دهد. تعداد پارامترهای کرنل در مدل کراس شما باید با تعداد پارامترهای فیلتر شما یکسان باشد.
۴. در فرآیند آموزش تصویر را به مدل بدهید و انتظار داشته باشید که لبه‌ها را به شما بدهد.
۵. وزنهای به دست آمده‌ی مدل کراس آموزش دیده را نمایش دهید.
۶. این وزنها را به کتابخانه سیگنال بدهید و خروجی آن به همراه تصویر اصلی؛ خروجی مدل آموزش دیده و لبه‌های اولیه را نمایش دهید

تکه برنامه زیر را ملاحظه فرموده و به پرسش‌های پس از آن پاسخ دهید.

```
1  prewitt_kernel = np.array([
2      [1, 0, -1],
3      [1, 0, -1],
4      [1, 0, -1]], dtype = np.float32)
5  # prewitt_img = signal.convolve2d(img, prewitt_kernel, mode="
       same")
6  prewitt_img = signal.correlate2d(img, prewitt_kernel, mode="
       same")
7  model = tf.keras.models.Sequential([
8      tf.keras.layers.Conv2D(1,  3,
9          input_shape=(img.shape[0],img.shape[0]  ,1),
10         padding="same",)])
11
12 X = tf.reshape(img, (1, img.shape[0],img.shape[0]  ,1))
13 y = tf.reshape(prewitt_img, (1, img.shape[0],img.shape[0]  ,1))
14 model.compile(optimizer= tf.keras.optimizers.Adam(
       learning_rate=0.01), loss='mean_squared_error')
15
16 model.fit(X, y, epochs=2000,verbose=False)
17 model.weights[0].numpy().reshape(3,3)
18
19 # And the output of the last command is as follows:
20
21 array([[ 0.90973  , -0.053146, -0.85232 ],
22        [ 1.27405  , -0.099578, -1.16464 ],
23        [ 0.80344  ,  0.156150, -0.97381]], dtype=float32)
```

اگر به سؤالات تمرین در برنامه فوق پاسخ داده شده است، بخش مربوطه را توضیح دهید. اگر به نظرتان بخشی از برنامه مرتبط با یک سؤال هست اما اشتباه پاسخ داده شده، مشخص کنید. اگر این تکلیف را در سامانه ارسال کرده‌اید، فقط برای بند ۳ (بخش **کرنلی را بیابید**) از صورت سؤال، مشخص کنید جواب شما با این پاسخ چه تفاوت‌هایی دارد؛ و در بخشهایی که متفاوت هست، آیا جواب شما درست بوده است یا این پاسخ؟ یک اشکال کوچک در این تمرین وجود داشت، متوجه شدید؟

۵. (۴۰ نمره) بخش اصلی کد مربوط به شبکه‌های مولد هم‌آورد در ادامه آورده شده است. کلیات روش کار این شبکه‌ها به همراه دو تابع زیر را توضیح دهید. به خط سوم و چرایی صفر یا یک بودن y_real, y_fake, y_gan هم در توضیح خود اشاره فرمایید.

```
1  def define_gan(generator, discriminator):
2    # make weights in the discriminator not trainable
3    discriminator.trainable = False
4    # connect them
5    model = Sequential()
6    # add generator
7    model.add(generator)
8    # add the discriminator
9    model.add(discriminator)
10   # compile model
11   model.compile(loss='binary_crossentropy', optimizer='adam')
12   return model
13 # train the generator and discriminator
14 def train(g_model, d_model, gan_model, latent_dim, n_epochs
     =10000, n_batch=128, n_eval=2000):
15   half_batch = int(n_batch / 2)
16   # manually enumerate epochs
17   for i in range(n_epochs):
18     # prepare real samples
19     x_real, y_real = generate_real_samples(half_batch)
20     # prepare fake examples
21     x_fake, y_fake = generate_fake_samples(g_model, latent_dim
         , half_batch)
22     # update discriminator
23     d_model.train_on_batch(x_real, y_real)
24     d_model.train_on_batch(x_fake, y_fake)
25     # prepare points in latent space as input for the
         generator
26     x_gan = generate_latent_points(latent_dim, n_batch)
27     # create inverted labels for the fake samples
28     y_gan = ones((n_batch, 1))
29     # update the generator via the discriminator's error
30     gan_model.train_on_batch(x_gan, y_gan)
```