به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام و نام خانوادگی	حسام اسداله‌زاده – مسعود طهماسبی
شماره دانشجویی	810198346 – 810198429
تاریخ ارسال گزارش	۱۴۰۱.۰۹.۰۳

فهرست

شکل‌ها

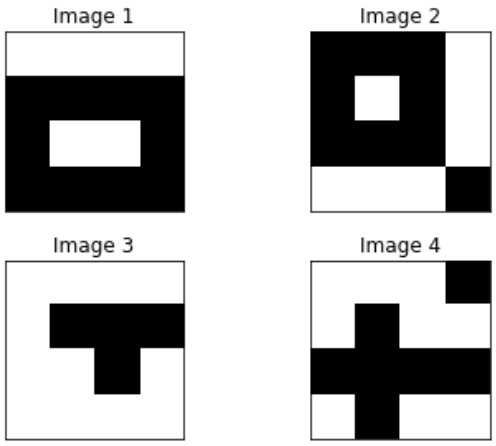
جدول‌ها

پاسخ 1. تاثیر تغییر رزولوشن در طبقه‌بندی در شبکه CNN

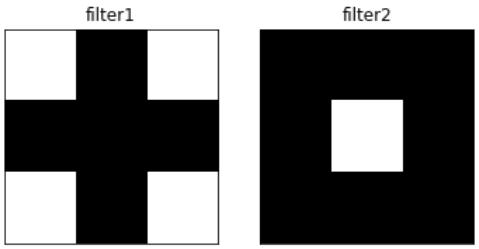
۱-۱- دست‌گرمی

پس از حل این مسئله به صورت دستی، کد مربوط به این مسئله را نیز پیاده‌سازی کردیم:

ابتدا تصاویر مسئله را به صورت آرایه‌های دوبعدی تعریف کردیم که مطابق مسئله برای پیکسل‌های سیاه مقدار ۱ و برای پیکسل‌های سفید عدد ۰ جایگذاری شده است. به همین ترتیب فیلترهای مربوط به دو کلاس مختلف نیز ساخته شده است.



شکل**1** . تصاویر ورودی مسئله دست‌گرمی



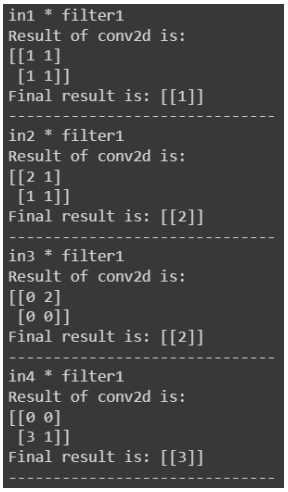
شکل**2** . تصاویر فیلترهای مسئله دست‌گرمی

حال عملیات کانولوشن توصیف شده در مسئله را با استفاده از تابع زیر پیاده‌سازی می‌کنیم:

```
[ ]  1 def conv(img, filter, bias=-2):
     2     res = convolve2d(img, filter, mode='valid')+bias
     3     print("Result of conv2d is:", res, sep='\n')    # Convolution
     4     res = np.maximum(0, res)                        # ReLU
     5     res = measure.block_reduce(res, (2, 2), np.max) # Max Pooling
     6     return res
```

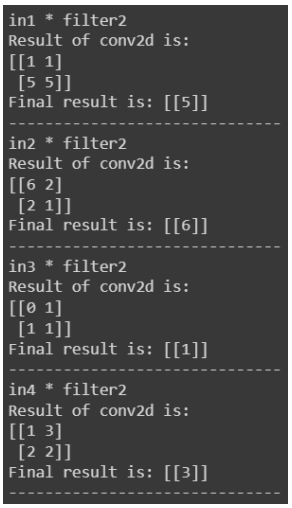
شکل 3. تابع کانولوشن دست‌گرمی

نتایج نهایی برای تصاویر و فیلترهای مختلف به شرح زیر است:



شکل 4. نتایج فیلتر اول

همانطور که مشاهده می‌شود، تصویر چهارم بهترین نتیجه را داشته است و تصاویر دوم و سوم به امتیاز مشابهی به دست آورده‌اند.

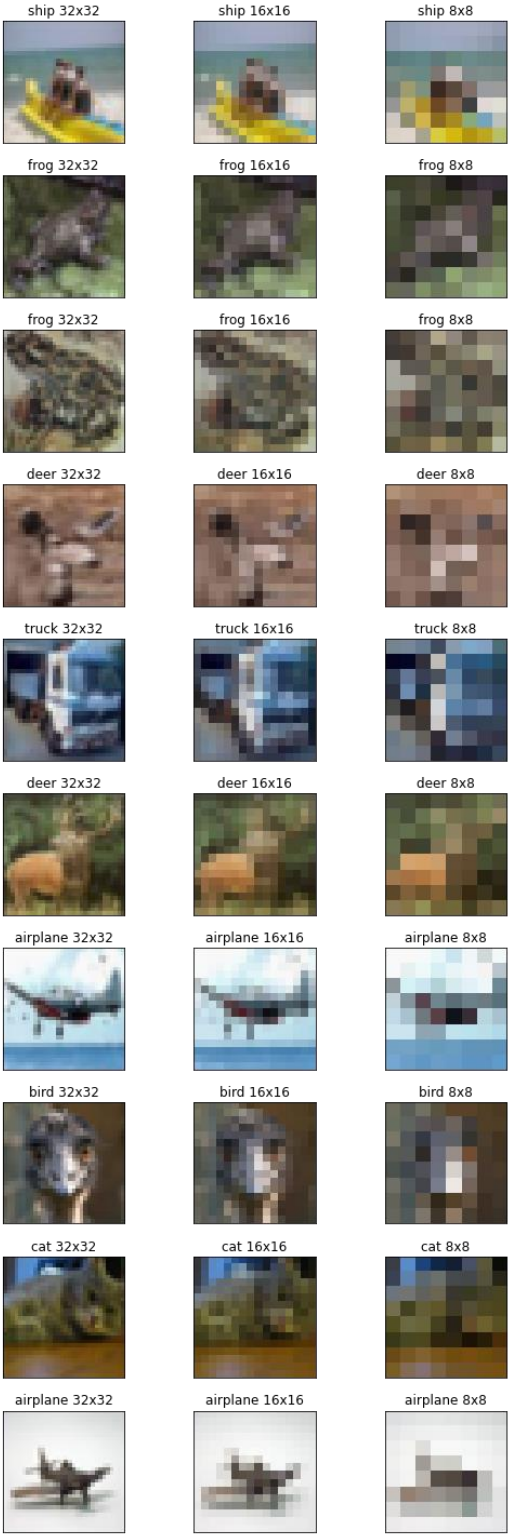


شکل 5. نتایج فیلتر دوم

برای این فیلتر نیز به وضوح امتیاز تصاویر اول و دوم بیشتر دو تصویر بعدی است که نشان می‌دهد این دو تصویر متعلق به یک کلاس (کلاس اول) تعلق دارند.

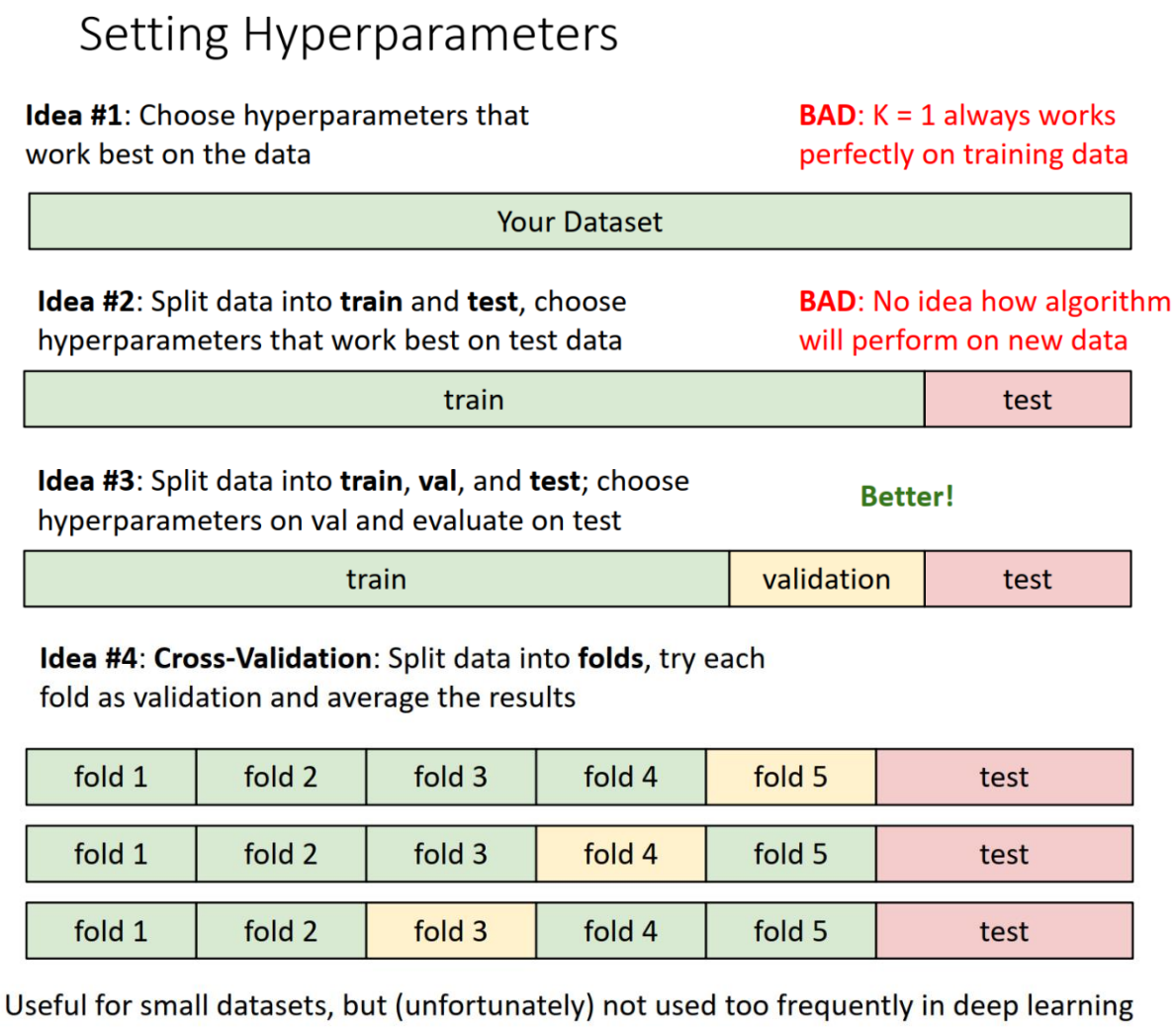
۱-۲- طبقه‌بندی تصاویر CIFAR-10

الف) ده تصویر تصادفی (نظیر به نظیر) در سه استایل مختلف:



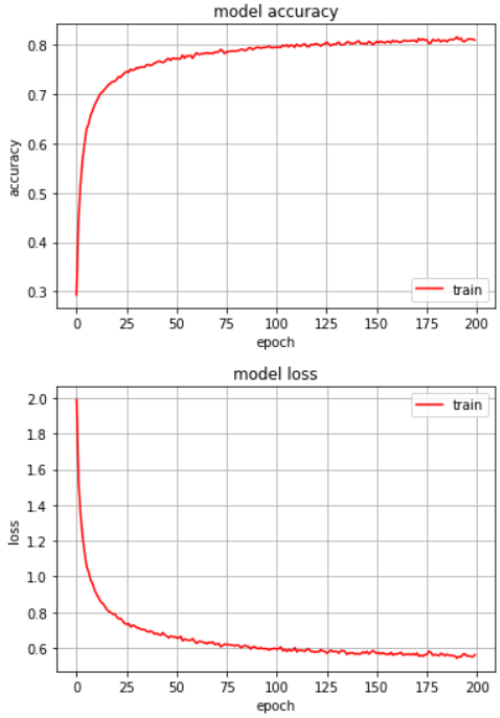
شکل 6. تصاویر **CIFAR-10** در رزولوشن‌های مختلف

ب) برای تعیین مقدار هایپرپارامترهای مدل یا ارزیابی معماری‌های مختلف برای حل یک مسئله‌ی خاص نیاز داریم که دقت مدل و توانایی generalization و تعمیم آن را روی داده‌های غیرتکراری (unseen) به دست آوریم. برای این منظور باید کل دیتاست خود را به سه بخش آموزش، تست و ارزیابی تقسیم نماییم. در حالت کلی ۴ روش مختلف برای این منظور وجود دارد:



همانطور که در تصویر مشخص است، بهترین روش برای ارزیابی، استفاده از روش K-fold cross validation می‌باشد که در این روش ابتدا کل داده را به دو قسمت آموزش و تست تقسیم می‌کنیم و سپس خود داده آموزش را به دو بخش ارزیابی و آموزش تقسیم می‌نماییم. با استفاده از این روش می‌توانیم هم به تعیین مقدار هایپرپارامترها و دقت و توانایی مدل بپردازیم و هم داده‌ی غیرتکراری واقعی در مجموعه تست داریم که بتوانیم قدرت generalization مدل نهایی را ارزیابی کنیم.

ج) روش TOTV:



شکل 7. نمودار تغییرات دقت (**accuracy**) و **loss** برای روش **TOTV** رزولوشن **32x32**

```
Epoch 200/200
391/391 [==============================] - 5s 12ms/step - loss: 0.5615 - accuracy: 0.8093
```

	precision	recall	f1-score	support
airplane	0.80	0.77	0.78	1000
automobile	0.93	0.85	0.88	1000
bird	0.67	0.65	0.66	1000
cat	0.59	0.60	0.60	1000
deer	0.70	0.76	0.73	1000
dog	0.67	0.73	0.70	1000
frog	0.71	0.86	0.78	1000
horse	0.84	0.76	0.80	1000
ship	0.87	0.81	0.84	1000
truck	0.88	0.81	0.84	1000
accuracy			0.76	10000
macro avg	0.77	0.76	0.76	10000
weighted avg	0.77	0.76	0.76	10000

شکل 8. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TOTV** رزولوشن **32x32**

	precision	recall	f1-score	support
airplane	0.59	0.37	0.46	1000
automobile	0.95	0.20	0.32	1000
bird	0.38	0.39	0.39	1000
cat	0.19	0.74	0.30	1000
deer	0.38	0.43	0.40	1000
dog	0.30	0.46	0.36	1000
frog	0.59	0.10	0.16	1000
horse	0.91	0.18	0.31	1000
ship	0.50	0.55	0.52	1000
truck	0.68	0.12	0.21	1000
accuracy			0.35	10000
macro avg	0.55	0.35	0.34	10000
weighted avg	0.55	0.35	0.34	10000

شکل 9. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TOTV** رزولوشن **16x16**

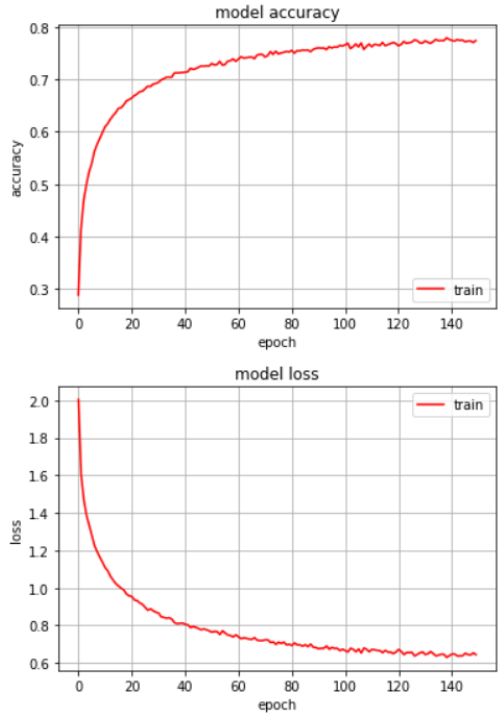
	precision	recall	f1-score	support
airplane	0.33	0.26	0.29	1000
automobile	0.86	0.01	0.02	1000
bird	0.21	0.18	0.20	1000
cat	0.15	0.77	0.26	1000
deer	0.24	0.28	0.26	1000
dog	0.25	0.10	0.14	1000
frog	0.33	0.09	0.14	1000
horse	1.00	0.01	0.01	1000
ship	0.30	0.44	0.36	1000
truck	0.32	0.01	0.02	1000
accuracy			0.21	10000
macro avg	0.40	0.21	0.17	10000
weighted avg	0.40	0.21	0.17	10000

شکل 10. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TOTV** رزولوشن **8x8**

د) روش TVTV:

	precision	recall	f1-score	support
airplane	0.80	0.77	0.78	1000
automobile	0.93	0.85	0.88	1000
bird	0.67	0.65	0.66	1000
cat	0.59	0.60	0.60	1000
deer	0.70	0.76	0.73	1000
dog	0.67	0.73	0.70	1000
frog	0.71	0.86	0.78	1000
horse	0.84	0.76	0.80	1000
ship	0.87	0.81	0.84	1000
truck	0.88	0.81	0.84	1000
accuracy			0.76	10000
macro avg	0.77	0.76	0.76	10000
weighted avg	0.77	0.76	0.76	10000

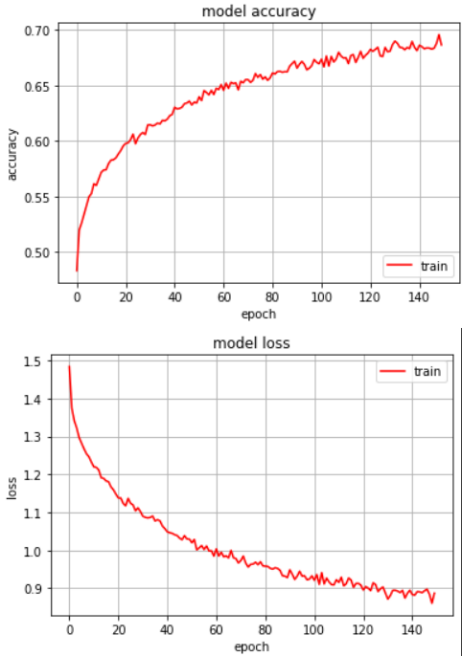
شکل 11. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TVTV** رزولوشن **32x32**



شکل 12. نمودار تغییرات دقت (**accuracy**) و **loss** برای روش **TVTV** رزولوشن **16x16**

	precision	recall	f1-score	support
airplane	0.79	0.66	0.72	1000
automobile	0.84	0.78	0.81	1000
bird	0.68	0.53	0.60	1000
cat	0.48	0.56	0.52	1000
deer	0.68	0.60	0.63	1000
dog	0.56	0.61	0.58	1000
frog	0.70	0.80	0.75	1000
horse	0.77	0.72	0.74	1000
ship	0.77	0.82	0.79	1000
truck	0.70	0.82	0.75	1000
accuracy			0.69	10000
macro avg	0.70	0.69	0.69	10000
weighted avg	0.70	0.69	0.69	10000

شکل 13. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TVTV** رزولوشن **16x16**



شکل 14. نمودار تغییرات دقت (**accuracy**) و **loss** برای روش **TVTV** رزولوشن **8x8**

	precision	recall	f1-score	support
airplane	0.59	0.68	0.63	1000
automobile	0.66	0.80	0.72	1000
bird	0.55	0.46	0.50	1000
cat	0.44	0.36	0.40	1000
deer	0.59	0.48	0.53	1000
dog	0.53	0.38	0.44	1000
frog	0.56	0.74	0.64	1000
horse	0.68	0.63	0.65	1000
ship	0.68	0.67	0.68	1000
truck	0.56	0.66	0.60	1000
accuracy			0.59	10000
macro avg	0.58	0.59	0.58	10000
weighted avg	0.58	0.59	0.58	10000

شکل 15. مقادیر **precision** و **accuracy** و **F-1 score** برای روش **TVTV** رزولوشن **8x8**

جدول 1. نتایج کلی برای **TOTV** و **TVTV** رزولوشن‌های مختلف

CIFAR10 Dataset Resolution	**TOTV**			**TVTV**		
	Accuracy	**Precision**	**F1 Score**	**Accuracy**	**Precision**	**F1 Score**
32x32	**76%**	**77%**	**76%**	**76%**	**77%**	**76%**
16x16	**35%**	**55%**	**34%**	**69%**	**70%**	**69%**
8x8	**21%**	**40%**	**17%**	**59%**	**58%**	**58%**

پاسخ ۲ – آشنایی با معماری شبکه CNN

۲-۱- لود دیتاست مقاله

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step

x_train.shape, x_test.shape

((60000, 28, 28), (10000, 28, 28))
```

شکل 16. لود دیتاست **Fashion-MNIST**

شکل 17. چند تصویر از دیتاست **CIFAR-10**

۲-۲- انتخاب معماری

در مقاله‌ی معرفی شده برای سوال، پنج معماری مختلف معرفی شده که چهار معماری دارای لایه‌های کانولوشنی می‌باشند و معماری اول صرفا از لایه‌های fully connected تشکیل شده است. با توجه به اینکه در تمرین اول، معماری MLP مورد بررسی قرار گرفت، در این سوال دو معماری دوم و چهارم را پیاده‌سازی و بررسی می‌کنیم تا نتایج مقاله را تصدیق کنیم:

Architecture 2	Architecture 4
2 convolutional layers with (2 x 2) filter size and 2 fully connected layers	4 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:2×2 size,64 filters (4) POOL:2×2 size (5) DROPOUT: = 0.25 (6) CONV2D :2×2 size,64 filters (7) DROPOUT: = 0.25 (8) FC:64 Hidden Neurons (9) DROPOUT: = 0.25	(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) POOL:2×2 size (11) DROPOUT: = 0.25 (12) CONV2D :2×2 size,64 filters (13) DROPOUT: = 0.25 (14) FC:64 Hidden Neurons (15) DROPOUT: = 0.25

شکل 18. دو معماری انتخاب شده

```
model = keras.Sequential([
    keras.layers.Input((28, 28, 1)),
    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(.25),
    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.Dropout(.25),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(.25),
    keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy',
                       tf.keras.metrics.Precision(),
                       tf.keras.metrics.Recall()]
)
```

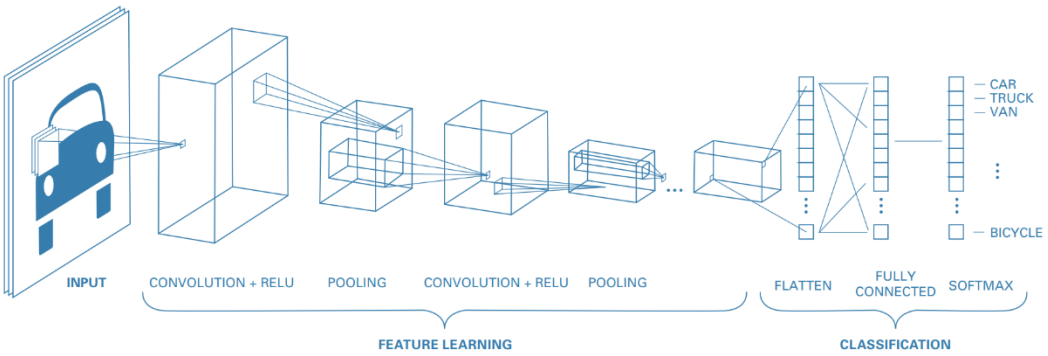
شکل 19. پیاده‌سازی معماری دوم با استفاده از کتابخانه **TensorFlow** و رابط **Keras**

```
model = keras.Sequential([
    keras.layers.Input((28, 28, 1)),

    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(.25),

    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(.25),

    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Dropout(.25),

    keras.layers.Conv2D(64, (2, 2), activation='relu'),
    keras.layers.Dropout(.25),

    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(.25),
    keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='RMSprop',
              loss='categorical_crossentropy',
              metrics=['accuracy',
                       tf.keras.metrics.Precision(),
                       tf.keras.metrics.Recall()]
)
```

شکل 20. پیاده‌سازی معماری چهارم با استفاده از کتابخانه **TensorFlow** و رابط **Keras**

۲-۳- توضیح لایه‌های مختلف معماری

همانطور که در توضیح معماری‌ها آورده شده، معماری دوم دارای ۲ لایه کانولوشن و ۲ لایه FC و معماری چهارم دارای ۴ لایه کانولوشن و ۲ لایه FC می‌باشد.



شکل 21. معماری کلی شبکه‌های کانولوشنی برای دسته‌بندی تصاویر

همانطور که در شکل 21 مشخص است، در حالت کلی شبکه‌های کانولوشنی در ابتدا از لایه‌های کانولوشن و pooling برای استخراج ویژگی استفاده می‌کنند و در لایه‌های آخر با استفاده از لایه‌های FC، دسته‌بندی تصاویر مختلف با استفاده از ویژگی‌های استخراج شده از لایه‌های قبلی را انجام می‌دهند. همچنین طبق تجربه اثبات شده که کانولوشن‌های 3×3 یا 2×2 که به طور متوالی انباشت[1] می‌شوند، می‌توانند به همان «میدان تأثیر» (receptive field) که کانولوشن‌های بزرگ‌تر ارائه می‌دهند دست یابند و این در حالی است که از نظر محاسباتی نیز کارایی بیشتری دارند.

همچنین لایه Pooling نیز در این معماری‌ها وجود دارد که مسئول کاهش سایز فضای ویژگی است. این کار با هدف کاهش قدرت محاسباتی مورد نیاز برای پردازش داده‌ها از طریق کاهش ابعاد، انجام می‌شود. علاوه بر این، برای استخراج ویژگی‌های «غالب» (Dominant) مفید است. لایه Max Pooling مقدار بیشینه را از قسمتی از تصویر بازمی‌گرداند که توسط کرنل پوشش داده شده است. Max Pooling کار «حذف نویز» را نیز انجام می‌دهد. این تجمع (Pooling)، همه فعال‌سازهای (Activations) نویزی را هم‌زمان رها می‌کند و همچنین، کار کاهش ابعاد را همراه با حذف نویز انجام می‌دهد.

در مورد کارایی لایه Dropout نیز در قسمت‌های بعدی توضیح ارائه خواهد شد.

[1] Stack

۲-۴- مقایسه نتایج دو معماری مختلف

	precision	recall	f1-score	support
T-shirt/top	0.86	0.89	0.87	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.85	0.91	0.88	1000
Dress	0.93	0.91	0.92	1000
Coat	0.87	0.89	0.88	1000
Sandal	0.99	0.98	0.98	1000
Shirt	0.81	0.73	0.77	1000
Sneaker	0.96	0.97	0.97	1000
Bag	0.99	0.97	0.98	1000
Ankle boot	0.96	0.97	0.97	1000
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

شکل 22. نتایج معماری دوم

برای معماری دوم، با استفاده از optimal parameterهای داده شده در مقاله، دقت ۹۲ درصد برای آموزش و تست به دست آمده که نتایج ما نیز این موضوع را تایید می‌کند.

	precision	recall	f1-score	support
T-shirt/top	0.85	0.83	0.84	1000
Trouser	1.00	0.97	0.98	1000
Pullover	0.83	0.87	0.85	1000
Dress	0.87	0.92	0.89	1000
Coat	0.83	0.81	0.82	1000
Sandal	0.99	0.96	0.97	1000
Shirt	0.68	0.66	0.67	1000
Sneaker	0.94	0.97	0.96	1000
Bag	0.98	0.98	0.98	1000
Ankle boot	0.96	0.96	0.96	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

شکل 23. نتایج معماری چهارم

برای معماری چهارم، با استفاده از optimal parameterهای داده شده در مقاله، دقت حدود ۹۲-۹۳ درصد برای آموزش و تست به دست آمده که نتایج ما نیز بسیار نزدیک به این مقادیر می‌باشند.

۲-۵- مقایسه نتایج استفاده از بهینه‌سازهای مختلف

نتایج استفاده از بهینه‌ساز Adam برای معماری دوم و بهینه‌ساز RMSprop برای معماری چهارم در قسمت ۲-۴ نمایش داده شد. حال به نمایش نتایج آموزش دو معماری با استفاده از بهینه‌ساز SGD می‌پردازیم:

	precision	recall	f1-score	support
T-shirt/top	0.82	0.84	0.83	1000
Trouser	0.99	0.97	0.98	1000
Pullover	0.82	0.81	0.82	1000
Dress	0.89	0.89	0.89	1000
Coat	0.78	0.88	0.82	1000
Sandal	0.98	0.95	0.97	1000
Shirt	0.73	0.63	0.68	1000
Sneaker	0.93	0.97	0.95	1000
Bag	0.96	0.97	0.97	1000
Ankle boot	0.96	0.95	0.96	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

شکل 24. نتایج معماری دوم با استفاده از بهینه‌ساز **SGD**

	precision	recall	f1-score	support
T-shirt/top	0.73	0.80	0.76	1000
Trouser	0.98	0.94	0.96	1000
Pullover	0.64	0.64	0.64	1000
Dress	0.76	0.88	0.81	1000
Coat	0.61	0.71	0.66	1000
Sandal	0.96	0.93	0.94	1000
Shirt	0.47	0.29	0.36	1000
Sneaker	0.90	0.93	0.91	1000
Bag	0.94	0.94	0.94	1000
Ankle boot	0.93	0.94	0.93	1000
accuracy			0.80	10000
macro avg	0.79	0.80	0.79	10000
weighted avg	0.79	0.80	0.79	10000

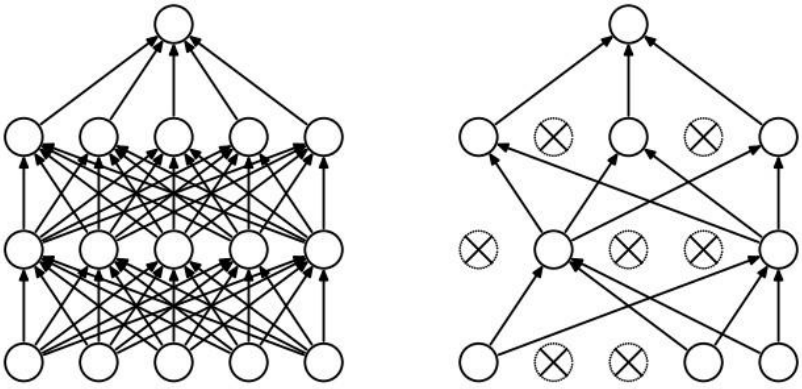
شکل 25. نتایج معماری چهارم با استفاده از بهینه‌ساز **SGD**

همانطور که مشخص است نتایج هر دو معماری پس از استفاده از بهینه‌ساز SGD ضعیف‌تر از بهینه‌ساز Adam و RMSprop بوده است. به طوری که کاهش ۳ درصدی دقت برای معماری دوم و کاهش ۹ درصدی دقت برای معماری چهارم رخ داده است. این موضوع به دلیل استفاده از مشتق دوم و روش‌های مبتنی بر Momentum در روش‌های نظیر Adam و RMSprop است که باعث می‌شود روند بهینه‌سازی با سرعت و دقت بیشتری انجام شود و نتایج بهتری به دست آید.

۲-۶- استفاده از Dropout

واژه Dropout به معنای کنار گذاشتن بخش‌هایی (units) از یک شبکه عصبی است. Dropout به این معنا است که در حین آموزشِ این نورون‌ها، از تعدادی از آن‌ها به صورت تصادفی چشم‌پوشی شود. چشم‌پوشی یعنی اینکه آن نورون‌های خاص، در مسیر رفت یا برگشت در نظر گرفته نمی‌شوند. یکی از علل استفاده از Dropout جلوگیری از overfitting است. در یادگیری ماشین، یکی از راه‌های جلوگیری از overfitting، رگولاریزاسیون است. رگولاریزاسیون با اضافه کردن یک پنالتی به تابع هزینه، overfitting را کاهش می‌دهد. با اضافه کردن این پنالتی، مدل به گونه‌ای آموزش داده می‌شود که وزن‌های ویژگی‌های

وابسته، آپدیت نمی‌شوند. Dropout یک رویکرد برای رگولاریزاسیون در شبکه‌های عصبی است که باعث کاهش یادگیری‌های تکراری میان نورون‌ها می‌شود. همچنین Dropout باعث می‌شود شبکه به نورون‌ها وابستگی نداشته باشد. به عبارت دیگر Dropout به طور تصادفی سلول‌های عصبی را از شبکه عصبی رها می‌کند که معادل آموزش شبکه‌های مختلف عصبی است. شبکه‌های مختلف به طور متفاوتی برتری خواهند داشت، به طوری که مدل ما برای تجزیه و تحلیل پیش بینی خوب خواهد بود. همچنین این روش به جهت کاهش محاسبات مورد نیاز برای انجام forward یا backward propagation نیز مفید است چراکه تعداد پارامترهای شبکه را کاهش می‌دهد.



شکل 26. استفاده از **Dropout** در شبکه‌های عصبی