به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین دوم

نام و نام خانوادگی	آناهیتا هاشم زاده – پرهام بیچرانلو
شماره دانشجویی	۸۱۰۱۰۰۵۰۲ - ۸۱۰۱۰۰۳۰۳
تاریخ ارسال گزارش	۱۴۰۱.۰۹.۰۲

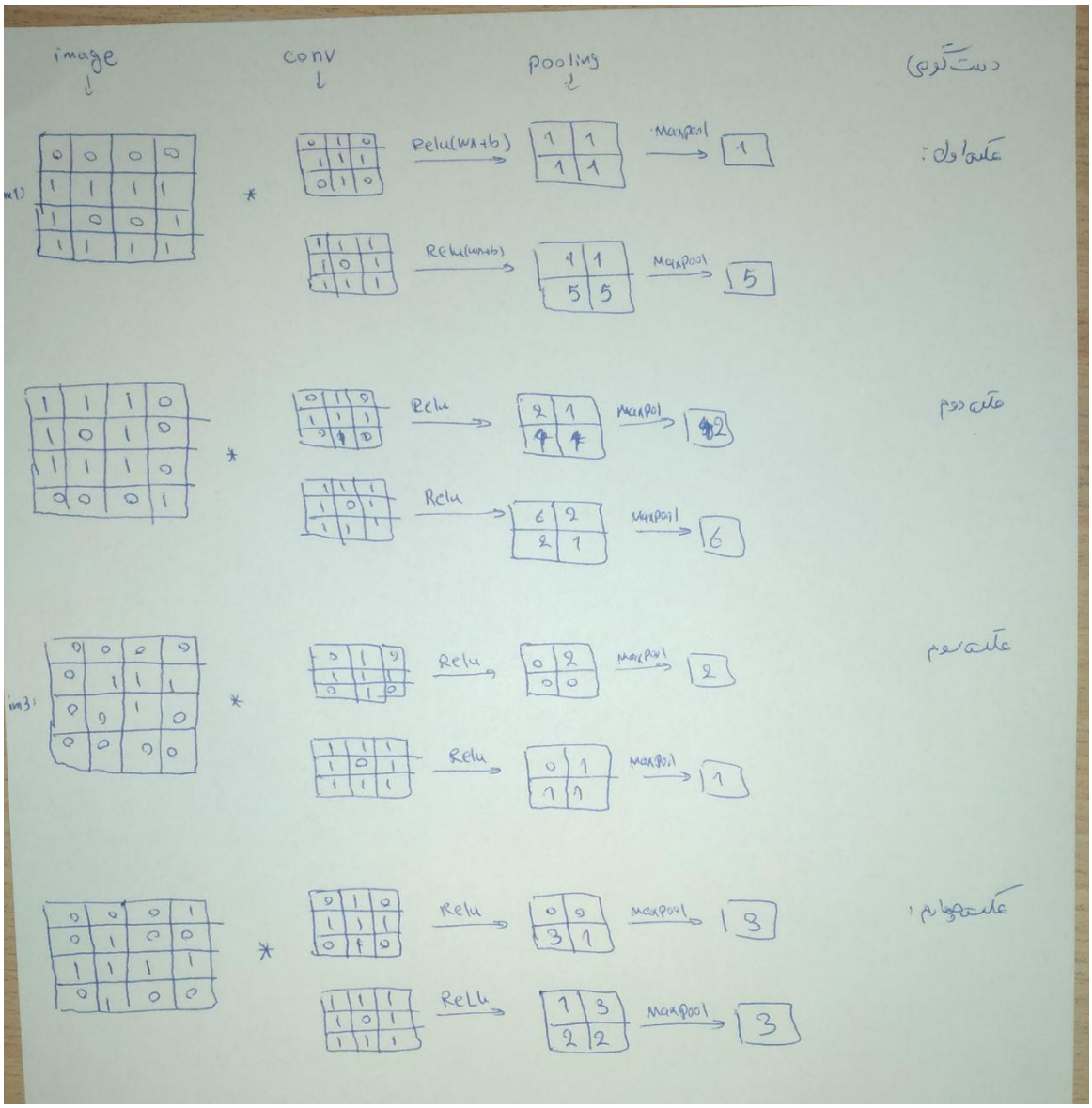
فهرست

شکل‌ها

جدول‌ها

پاسخ ۱. تاثیر تغییر رزولوشن در طبقه بندی در شبکه CNN

۱-۱. دست گرمی



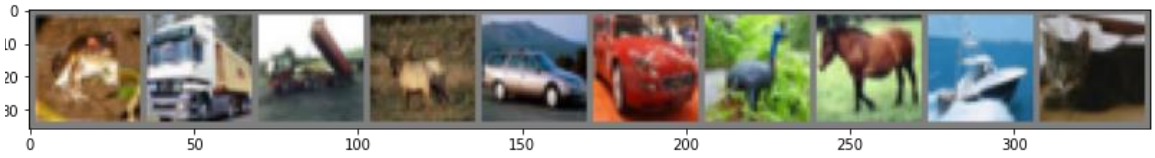
شکل ۱. رسم عملیات کانولوشن و مکس پولینگ

۱-۱. الف.

این سوال با کتابخانه پایتورچ و در محیط کولب پیاده سازی شده است. از GPU مدل T4 برای اجرا استفاده شده است.

برای قسمت الف دیتا را با استفاده از کتابخانه torchvision خواندیم. سپس با استفاده از DataLoader آن را به batch های با سایز ۶۴ تبدیل کردیم. برای تغییر رزولوشن ابتدا یک transform.Resize((x,y)) زدیم تا سایز را به ۱۶ یا ۸ کاهش دهیم سپس دوباره با همین دستور به سایز ۳۲ در ۳۲ برگرداندیم. اگر به سایز اصلی بر نمی‌گرداندیم مدل خطا می‌داد چون سایزش در لایه‌های میانی صفر می‌شد!

تصویر ۱۰ شکل برای هر سه رزولوشن در زیر آورده شده است.



شکل ۲. ده شکل اول برای رزولوشن ۳۲*۳۲

شکل ۳. ده شکل اول برای رزولوشن ۱۶*۱۶

شکل ۴. ده شکل اول برای رزولوشن ۸*۸

۲-۱. ب.

- Random: نمونه گیری را به صورت تصادفی انجام می‌دهیم. به اندازه درصد تعیین شده برای داده‌های train, validation, test از داده به صورت تصادفی انتخاب می‌شوند.

عیب: اگر کلاس‌ها نامتوازن باشند مثلا ممکن است داده train فقط از یک کلاس باشد و داده تست از کلاس دیگر. درحالی که باید از هر دو کلاس در train و test وجود داشته باشد تا مدل سوگیری نداشته باشد.

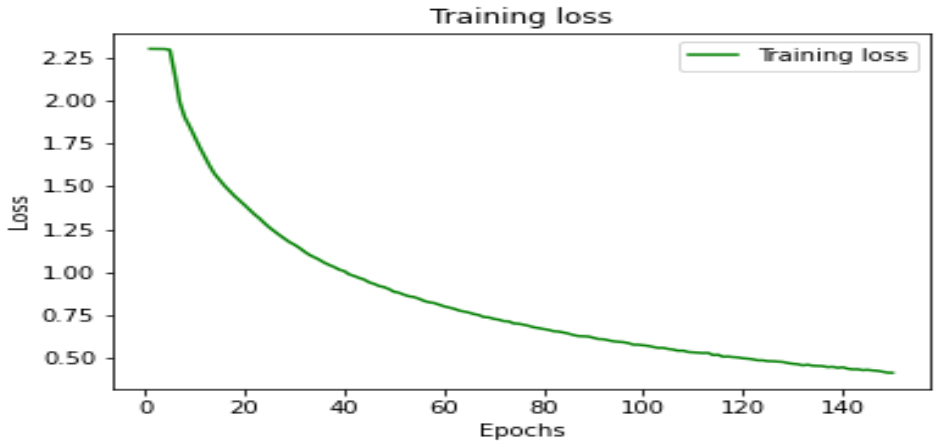
- Stratified: در این روش از هر کلاس به اندازه میزان درصدش در کل داده برای train و test و validation به صورت تصادفی انتخاب می‌شود. یعنی داده‌های هر کلاس را جدا کرده سپس مثلا ۸۰ درصد از هر کلاس را به train می‌دهیم ۱۰ درصد به کلاس validation و ۱۰ درصد به کلاس test می‌دهیم.
- Cross-Validation: در این روش مدل k بار آموزش داده شده و تست می‌شود. داده را به k قسمت تبدیل کرده در هر بار آموزش مدل m قسمت را به train می‌دهیم. n قسمت را به validation می‌دهیم و k-(n+m) قسمت را به مجموعه test می‌دهیم.

این روش بهترین روش جداکردن داده است. چون مدل در معرض توزیع‌های مختلف داده‌ها قرار می‌گیرد. در عین حال توزیع کلی کلاس‌ها را هم لحاظ می‌کند.

۳-۱. ج.

در روش TOTV باید با تصاویر سایز اصلی یعنی ۳۲*۳۲ مدل را آموزش داده سپس برای تست تمام رزولوشن‌ها(۳۲*۳۲و۱۶*۱۶و۸*۸) استفاده می‌کنیم.

نمودار loss-train در شکل زیر آمده است. که نشان می‌دهد روند آموزش منطقی است و به آرامی در حال کاهش خطا است.



شکل ۵. نمودار Training loss برای شبکه ۳۲*۳۲

متریک‌های بدست آمده از پیاده سازی روش TOTV در جدول زیر گزارش شده اند.

CIFAR10 Dataset Resolution	TOTV		
	Accuracy	Precision	F1 Score
32x32	75%	0.8125	0.8125
16x16	41%	0.475	0.475
8x8	24%	0.3125	0.3125

جدول ۱. مقدار متریک‌های روش TOTV

طبیعی است که این مدل برای رزولوشن ۳۲*۳۲ خوب جواب می‌دهد ولی برای روزولشن‌های دیگر یعنی ۱۶*۱۶ و ۸*۸ دقت کمتری داشته باشد چون مدل داده‌های رزولوشن‌های دیگر را ندیده است.

Accuracy for different classes - TOTV			
Class	32*32	16*16	8*8
Plane	71.6%	44.6%	33.9%
Car	82.3%	28.3%	1.5%
Bird	62.3%	50.5%	30.9%
Cat	63.8%	77.6%	82.6%
Deer	70.5%	29.0%	23.0%
Dog	62.1%	35.4%	14.8%
Frog	82.4%	12.1%	3.8%
Horse	80.7%	30.1%	6.0%
Ship	84.4%	58.8%	38.7%
Truck	79.9%	10.9%	0.3%

جدول ۲. دقت برای هر کلاس روش TOTV

به نظر می‌رسد که با کاهش رزولوشن دقت بعضی کلاس‌ها محسوس‌تر کاهش پیدا می‌کند مثل کلاس car, Deer, Frog, Horse, Truck.

و نکته جالب این است که برای کلاس Cat با کاهش رزولوشن دقت بهتر شده است! که کمی عجیب است چون جزئیتای رو از دست دادیم و باید دقت کمتر شود.

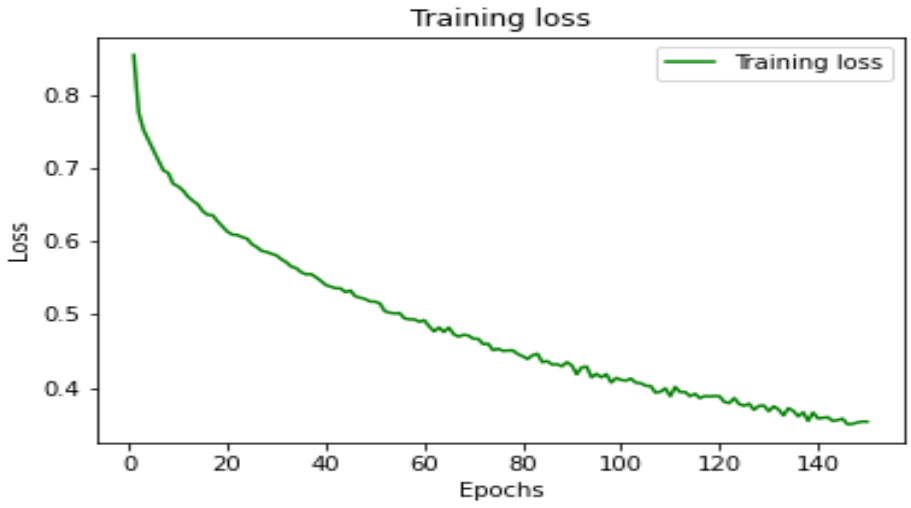
معماری شبکه و تعداد پارمترها:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 30, 30]	896
Conv2d-2	[-1, 32, 28, 28]	9,248
Conv2d-3	[-1, 32, 26, 26]	9,248
MaxPool2d-4	[-1, 32, 13, 13]	0
Dropout-5	[-1, 32, 13, 13]	0
Conv2d-6	[-1, 64, 11, 11]	18,496
Conv2d-7	[-1, 64, 9, 9]	36,928
Conv2d-8	[-1, 64, 7, 7]	36,928
MaxPool2d-9	[-1, 64, 3, 3]	0
Dropout-10	[-1, 64, 3, 3]	0
Linear-11	[-1, 512]	295,424
Dropout-12	[-1, 512]	0
Linear-13	[-1, 10]	5,130

۴-۱. د.

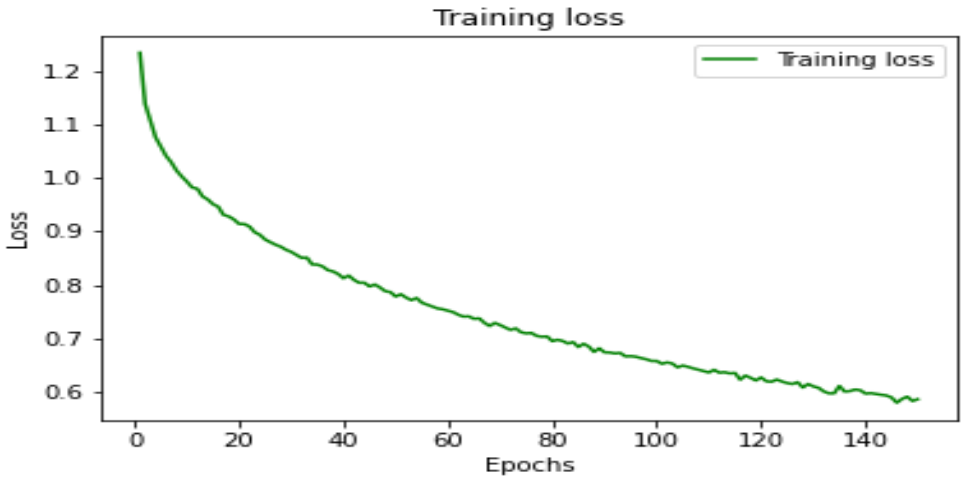
در روش TVTV باید برای هر رزولوشن یک مدل جداگانه آموزش داد و سپس داده تست رزولوشن‌های مختلف را به مدل نظیرش می‌دهیم تا دقت را بدست آوریم.

نمودار loss-train برای شبکه ترین شده با دیتای ۱۶*۱۶ در شکل زیر آمده است. که نشان می‌دهد روند آموزش منطقی است و به آرامی در حال کاهش خطا است.



شکل ۶. نمودار Training loss برای شبکه ۱۶*۱۶

نمودار loss-train برای شبکه ترین شده با دیتای ۸*۸ در شکل زیر آمده است. که روند کاهش خطا در این نمودار هم منطقی است.



شکل ۷. نمودار Training loss برای شبکه ۸*۸

متریک‌های بدست آمده از پیاده سازی روش TVTV در جدول زیر آورده شده است.

CIFAR10 Dataset Resolution	**TVTV**		
	Accuracy	Precision	F1 Score
32x32	75%	0.8125	0.8125
16x16	70%	0.75	0.75
8x8	59%	0.5	0.5

جدول ۳. مقدار متریک‌های روش **TVTV**

همانطور که می‌بینید این روش دقت خیلی بهتری از روش قبلی می‌دهد چون از مدل توقع نداریم با دیدن یک رزولوشن بتواند همه رزولوشن‌ها را پیش بینی کند. بلکه در این روش داده‌های train و test توزیع یکسانی دارند و همین باعث می‌شود دقت نزدیک دو برابر برای رزولوشن‌های ۱۶*۱۶ و ۸*۸ بهبود یابد.

Accuracy for different classes - TVTV			
Class	32*32	16*16	8*8
Plane	71.6%	68.5%	67.1%
Car	82.3%	81.0%	71.8%
Bird	62.3%	56.4%	41.5%

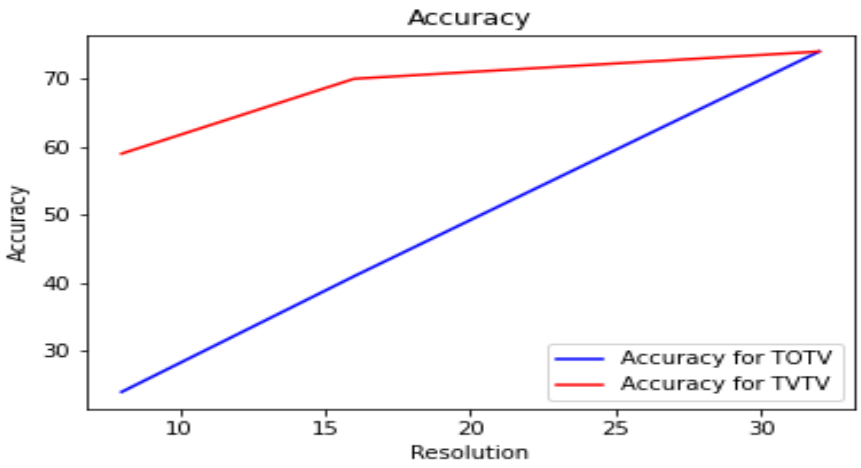
Cat	63.8%	52.5%	48.8%
Deer	70.5%	64.3%	50.1%
Dog	62.1%	64.7%	54.3%
Frog	82.4%	73.8%	65.1%
Horse	80.7%	78.2%	69.0%
Ship	84.4%	86.7%	66.5%
Truck	79.9%	81.0%	66.4%

جدول ۴. دقت برای هر کلاس روش **TVTV**

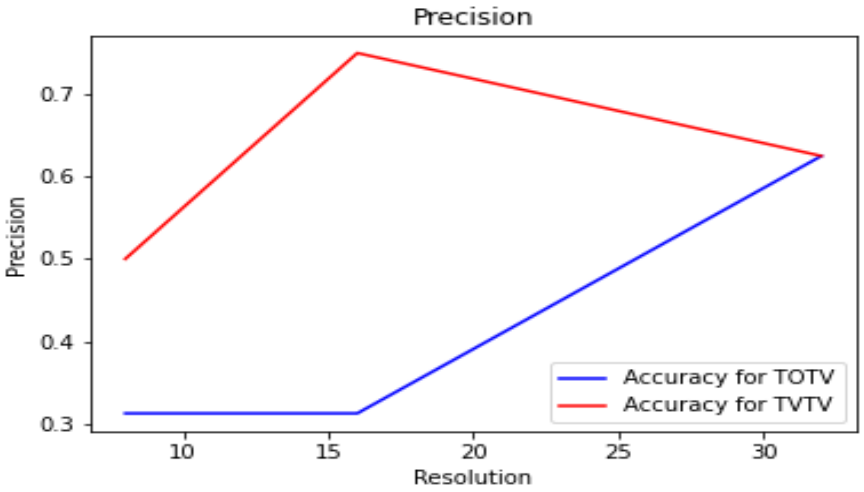
جمع بندی:

روش TVTV بسیار بهتر از روش TOTV عمل می‌کند. دلیل آن هم این است که در روش اول توزیع داده train و test یکسان است درحالی که در روش دوم فرق دارد.

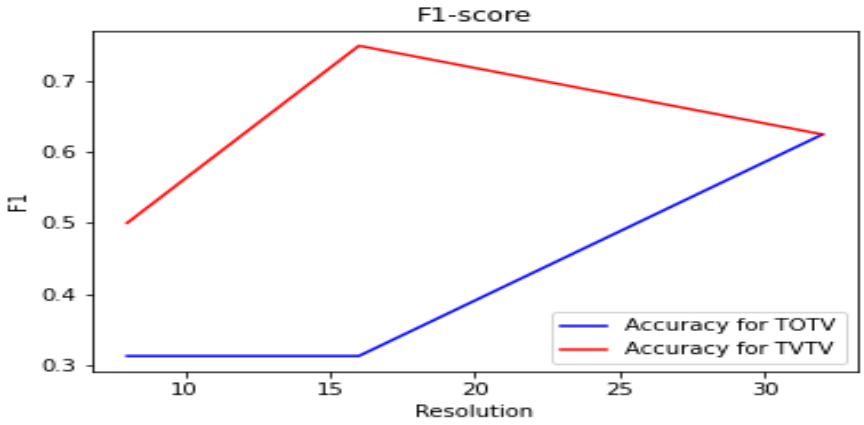
نمودار مقدار متریک‌های مختلف برای سه رزولوشن با دو روش در شکل‌های زیر آمده اند.



شکل ۸. مقایسه دقت در دو روش



شکل ۹. مقایسه precision در دو روش



شکل ۱۰. مقایسه F1_score در دو روش

سایر نکات:

- هایپر پارامترها در مقاله آورده نشده. با کمی سعی خطا سعی کردیم به مقادیر نسبتا مناسبی برسیم. که عبارتند از:
  - تعداد ایپاک: **150**
  - نرخ یادگیری: **0.001**
  - مومنتوم: **0.9**
- از بهینه ساز SGD استفاده کردیم.

- چون این جزئیات در مقاله آورده نشده نتایج بدست آمده با مقاله کمی فرق دارد اما نتیجه کلی مقاله با نتایج ما هم خوانی دارد.
- با کارهایی مثل بیشتر کردن تعداد ایپاک‌ها، batch normalization، Data augmentation می‌توانستیم دقت مدل را بیشتر هم کنیم. که چون در مقاله ذکر نشده بود و محدودیت پردازشی داشیم و تقریبا نزدیک دقت گزارش شده در مقاله را بدست آوردیم از این کارها صرف نظر کردیم.

پاسخ ۲. آشنایی با معماری شبکه CNN

۲-۱. لود دیتاست مقاله

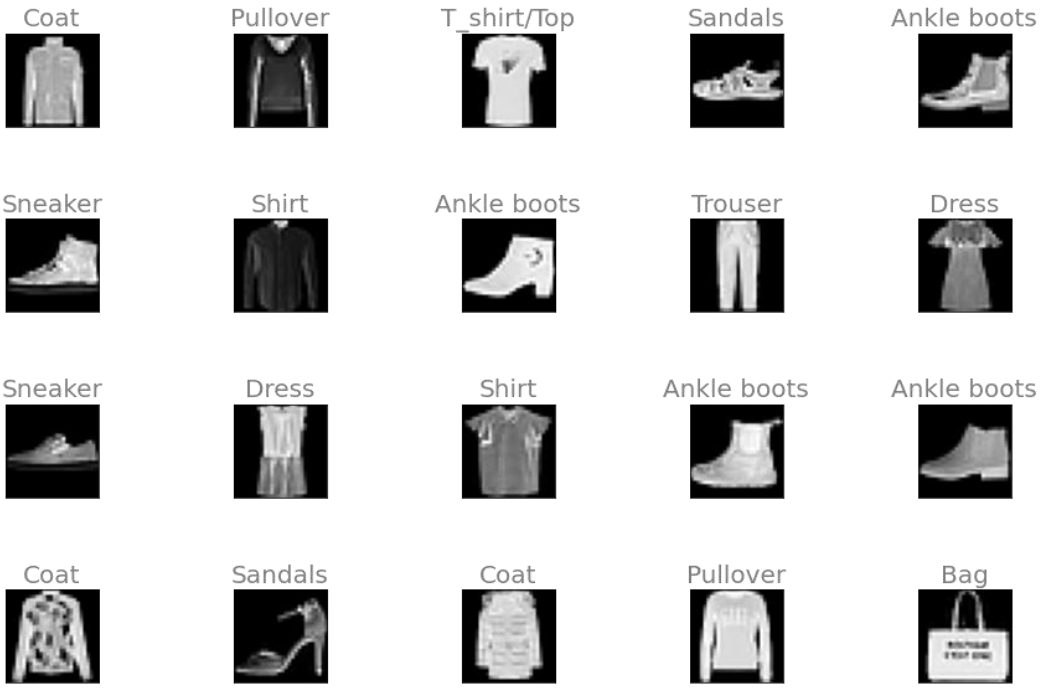
در ابتدا دیتاست Fashin-MNIST را از طریق دستور torchvision.datasets.FashionMNIST در کتابخانه pytorch لود میکنیم. داده های آموزش ۶۰۰۰۰ و داده های تست ۱۰۰۰۰ داده را شامل می شوند، که در ادامه همانطور که در صورت سوال گفته شده داده آموزش را split کرده و ۸۰ درصد آن برای داده آموزش و ۲۰ درصد نیز برای داده validation در نظر گرفته می شود. در نهایت ۴۸۰۰۰ داده آموزش، ۱۲۰۰۰ داده validation و ۱۰۰۰۰ داده تست خواهیم داشت. این دیتاست دارای ۱۰ کلاس می باشد که در ادامه چند نمونه از داده های آموزش نشان داده شده است.

جدول ۵. split داده ها

Train	48000
Validation	12000
Test	10000

جدول ۶. کلاس های داده

id	classes
0	**T_shirt/Top**
1	**Trouser**
2	**Pullover**
3	**Dress**
4	**Coat**
5	**Sandals**
6	**Shirt**
7	**Sneaker**
8	**Bag**
9	**Ankle boots**



شکل۱۱- چند نمونه از داده ها به همراه کلاس آنها

۲-۲. انتخاب معماری

در مقاله، ۵ معماری متفاوت استفاده شده که طبق صورت سوال دو معماری، معماری های شماره ۲ و شماره ۳ انتخاب و پیاده سازی شده است.

جدول ۷. Net hyper parameters

Network Used Hyper Parameters			
Batch Size			128
Epochs			50
Input			28*28*1
Output			10
Activation Functions			ReLU , SoftMax
➢ Arc2	➢ Model1	Loss Function	Cross Entropy
		Optimizer	SGD
		Learning rate	0.001
		momentum	0.9
		Max Train Accuracy reached	
		Mx Val Accuracy reached	

	➢ **Model2**	Loss Function	Cross Entropy
		Optimizer	Adam
		Learning rate	0.001
		Max Train Accuracy reached	
		Max Val Accuracy reached	
➢ **Arc3**	➢ **Model1**	Loss Function	Cross Entropy
		Optimizer	SGD
		Learning rate	0.001
		momentum	0.9
		Max Train Accuracy reached	
		Mx Val Accuracy reached	
	➢ **Model2**	Loss Function	L1
		Optimizer	Adam
		Learning rate	0.001
		Max Train Accuracy reached	
		Mx Val Accuracy reached	

۲-۳. توضیح لایه های مختلف معماری

معماری سوم نسبت به معماری دوم یک لایه Conv بیشتر دارد و از آنجایی که معماری عمیق تر می شود، لایه ای که classification را نمایش می دهد جزییات بیشتری نمایش می دهد. در واقع، لایه کانولوشن اضافه شده باعث دید بهتر و receptive field بیشتر شده و فیچرهای بهتر را برای طبقه بندی استخراج می کند. همچنین باعث پیچیدگی بیشتر مدل می شود و در نتیجه بهتر مدل می کند.

جدول ۸. Net Structure

Architecture	layers	InSize	OutSize	KernelSize
➢ **Arc2**	Conv1	1	64	2
	MaxPool2	64	64	2
	Dropout	64	64	0.25
	Conv2	64	64	2
	Dropout	64	64	P= 0.25
	FC1	9216	64	
	FC2	64	10	
➢ **Arc2**	Conv1	1	64	2
	MaxPool2	64	64	2
	Dropout	64	64	P= 0.25
	Conv2	64	64	2
	Conv3	64	64	2
	Dropout	64	64	P= 0.25
	FC1	1600	64	
	FC2	64	10	

هر دو معماری دارای لایه های کانولوشنال و همچنین pooling و dropout و fully connected هستند، تنها تفاوت در این دو معماری وجود یک لایه کانولوشن بیشتر است، به عبارتی معماری شماره ۲ دارای دو لایه کانولوشن و دو لایه fully connected است در حالی که معماری شماره ۳ دارای سه لایه کانولوشن و دو لایه fully connected می باشد.

- جزییات معماری شماره ۲ به صورت زیر می باشد:

```
NET_arc2(
  (conv1): Conv2d(1, 64, kernel_size=(2, 2), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (drop): Dropout(p=0.25, inplace=False)
  (conv2): Conv2d(64, 64, kernel_size=(2, 2), stride=(1, 1))
  (fc1): Linear(in_features=9216, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```

- جزییات معماری شماره ۲ به صورت زیر می باشد:

```
NET_arc3(
  (conv1): Conv2d(1, 64, kernel_size=(2, 2), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=1, ceil_mode=False)
  (drop): Dropout(p=0.25, inplace=False)
  (conv2): Conv2d(64, 64, kernel_size=(2, 2), stride=(1, 1))
  (conv3): Conv2d(64, 64, kernel_size=(2, 2), stride=(1, 1))
  (fc1): Linear(in_features=1600, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=10, bias=True)
)
```

۴-۲. مقایسه نتایج دو معماری مختلف

در این بخش نتایج accuracy, f1 score و precision مربوط به هر کلاس به صورت جداگانه برای هر دو معماری با دو مدل متفاوت در جدول زیر آورده شده است.

در معماری شماره دو بهترین دقت مربوط به داده آموزش با Adam optimizer ۹۳.۱۶ درصد و برای داده تست ۹۱.۶۴ می باشد، همچنین در معماری شماره سه با Adam optimizer بهترین دقت مربوط به داده آموزش برابر ۸۹.۵ درصد و برای داده تست برابر ۹۰.۱۸ درصد می باشد.

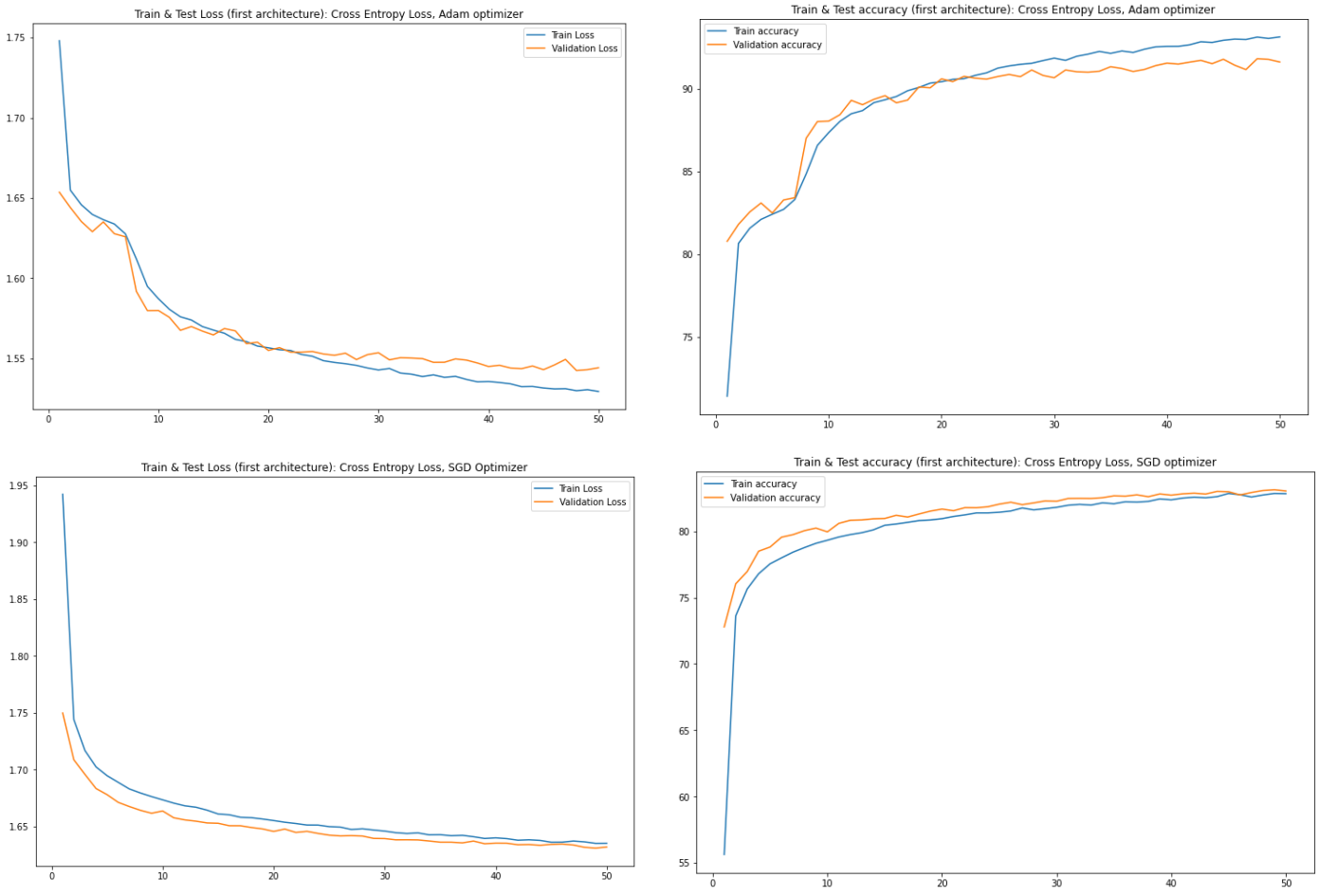
جدول ۹. نتایج accuracy, f1 score و precision مربوط به هر کلاس

Architecture	State	Class	Accuracy	F1 score	Precision
Arc2	Model1: (cross entropy, adam optimizer)	0	85.5 %	0.85	0.82
		1	97.6 %	0.98	0.99
		2	85.4 %	0.84	0.81
		3	93.1 %	0.90	0.88
		4	92.2 %	0.81	0.77
		5	97.8 %	0.98	0.98
		6	72.1 %	0.66	0.80
		7	98.0 %	0.96	0.95
		8	98.0 %	0.97	0.98
		9	95.0 %	0.97	0.97
	Model2: (cross entropy, SGD)	0	88.7 %	0.79	0.71
		1	95.9 %	0.97	0.98
		2	85.8 %	0.72	0.66
		3	91.6 %	0.87	0.83
		4	86.7 %	0.71	0.60
		5	93.8 %	0.93	0.93
		6	83.7 %	0.70	0.60
		7	95.9 %	0.91	0.90
		8	97.1 %	0.94	0.93
		9	94.5 %	0.94	0.94
Arc3	Model1: (cross entropy,	0	88.7 %	0.84	0.82
		1	95.9 %	0.98	0.98
		2	85.8 %	0.83	0.81

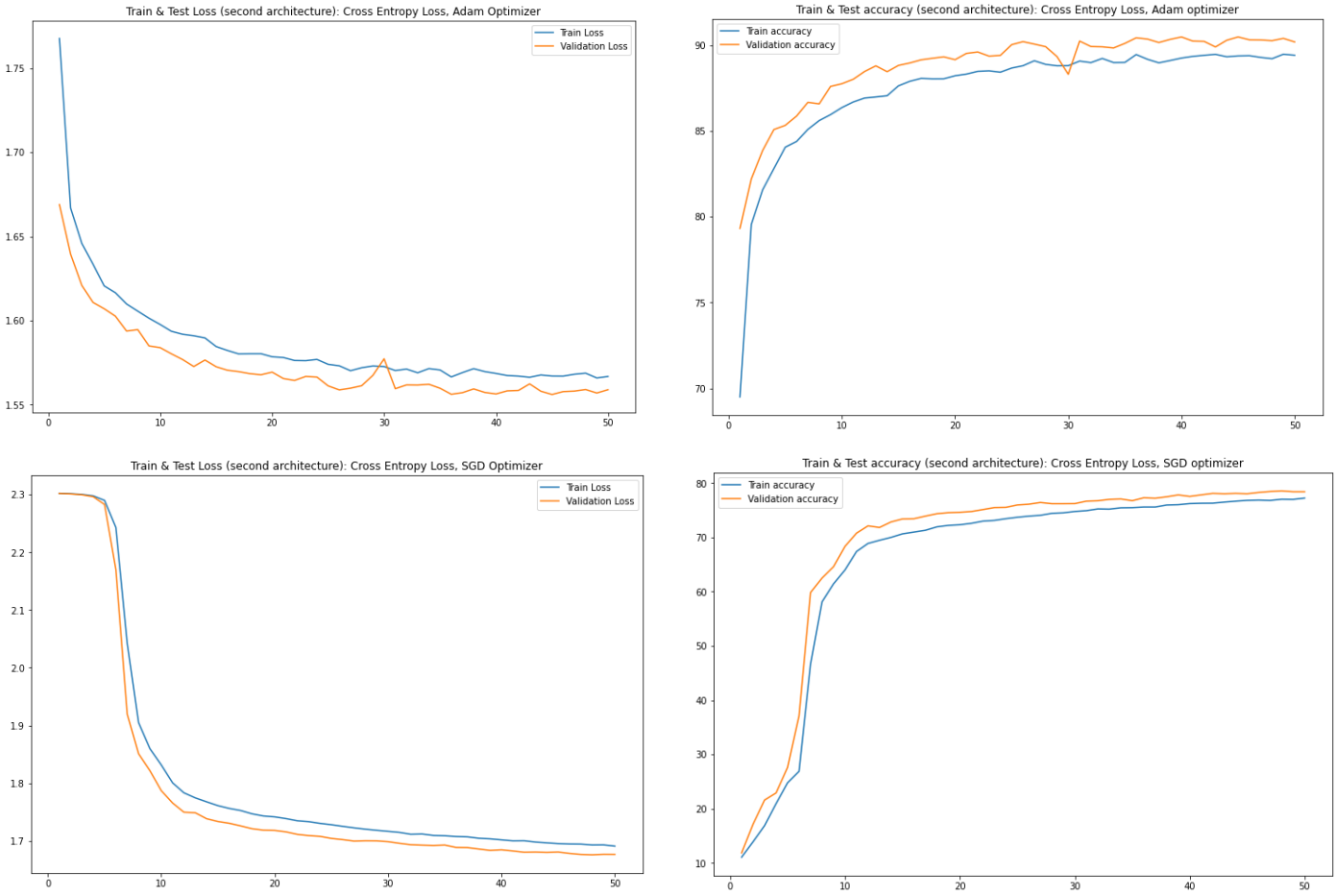
	adam optimizer)	3	91.6 %	0.89	0.87
		4	86.7 %	0.81	0.79
		5	93.8 %	0.97	0.97
		6	85.6 %	0.68	0.76
		7	95.9 %	0.95	0.94
		8	97.1 %	0.97	0.97
		9	94.5 %	0.96	0.96
	Model2: (cross entropy, SGD)	0	88.7 %	0.70	0.73
		1	95.9 %	0.86	0.87
		2	85.8 %	0.56	0.57
		3	91.6 %	0.72	0.66
		4	86.7 %	0.60	0.50
		5	93.8 %	0.75	0.71
		6	86.7 %	0.60	0.50
		7	95.9 %	0.67	0.56
		8	97.1 %	0.85	0.91
		9	94.5 %	0.86	0.88

۲-۵. مقایسه نتایج استفاده از بهینه سازهای مختلف

در این بخش هر دو مدل از دو معماری را با تابع هزینه cross entropy و optimizerهای متفاوت (Adam,SGD) آموزش داده و نمودار train loss و validation loss آنها و همچنین train accuracy و validation accuracy آنها بعد از ۵۰ ایپاک بررسی شده است.



شکل ۱۲. Accuracy & Loss of Arc2



شکل ۱۳. Accuracy & Loss of Arc3

با توجه به شکل فوق دیده می شود بعد از گذشت ۵۰ ایپاک دقت افزایش چشمگیری داشته، در تمامی مدل ها به بالاتر از ۸۰ درصد رسیده که در مدل اول دقت به ۹۱.۶ درصد نیز رسیده است و همچنین تقریبا بعد از ۴۰ ایپاک مقدار خطا هم برای داده های آموزش و هم ارزیابی بهبود آنچنانی نداشته است. در تمامی مدل ها نوساناتی در مقدار خطا دیده می شود ولی به طور کلی سیر نزولی نمودار طی می شود، که می توان دلیل آن را تقریب زدن هر batch از نمونه کلی دانست. همانطور که انتظار می رفت مدل با optimizer، Adam در هر دو معماری عملکرد بهتری دارد.

در ادامه ۱۵ داده به طور تصادفی از دادگان تست انتخاب شده و با استفاده از هر دو معماری کلاس آنها پیش بینی شده است:

شکل ۱۴. پیشبینی مدل معماری دوم از کلاس ۱۵ داده

شکل ۱۵. پیشبینی مدل معماری دوم از کلاس ۱۵ داده

۲-۶. استفاده از Dropout

تکنیک dropout از overfitting در شبکه عصبی جلوگیری میکند، چون در هنگام training برخی unitها را دور میریزد و در فرایند آموزش دخالت نمی دهد. به عبارت دیگر، در این روش یک سری از نودها به صورت رندوم انتخاب شده و غیر فعال می شوند که به مثابه افزودن نویز به مدل است که باعث robust شدن مدل و کم شدن وابستگی مدل به نواسانات می شود.