دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین ۳

علی صداقی

۹۷۵۲۱۳۷۸

سوال ۱)

الف) نمودار loss به ازای epoch بهینه‌سازهای مختلف مقایسه شده که درباره هریک می‌توان به نکات زیر اشاره کرد.

SGDNesterov: در واقع همان SGD همراه با Momentum است که به‌جای گذشته به آینده نگاه می‌کند. مشکل این روش این است که طول گام (learning rate) در همه جهات یکسان است و ممکن است در فیچرهایی که تاثیر زیادی روی کم شدن loss حرکت زیادی کنیم یا در جهت‌های مناسب حرکت کمی داشته باشیم. همه این موارد باعث می‌شود زمان همگرایی زیاد شود و در مدت تعیین شده نتوانیم به خوبی در نقطه Global Optimum همگرا شویم.

AdaGrad: این روش مشکل ثابت بودن طول قدم در جهات مختلف را حل می‌کند و با ارائه "نرخ یادگیری بر پارامتر" یا "نرخ یادگیری تطبیقی" باعث می‌شود در هر جهت طول قدم مناسبی داشته باشم. این کار باعث می‌شود در جهت‌های نامناسب حرکت کمی داشته باشیم و بیشتر حرکتمان به سمت نقطه min باشد. مشکلی که دارد در واقع این است که از کار مثبت Momentum بهره نبرده و نمی‌تواند مفهوم شتاب و سرعت را همزمان داشته باشد. در نتیجه عملکردش حتی از SGDNesterov بدتر است زیرا سریعا در نقطه Local Minima گیر می‌کند.

RMSprop: مشابه روش AdaGrad می‌باشد با این تفاوت که برای تعیین نرخ یادگیری تطبیقی از نوع خاصی از میانگین‌گیری استفاده می‌کند (EMA) این کار باعث می‌شود تنها به گرادیان حال حاضر نگاه نکنیم و با توجه به گرادیان نقاط گذشته و اکنون حرکت کنیم. در نتیجه احتمال پرش از Global Optimum کاهش می‌یابد. مشکل آن این است که از روش Momentum استفاده نمی‌کند. در نتیجه عملکردش از AdaGrad بهتر ولی از SGDNesterov بدتر است.

AdaDelta: بسیار شبیه به RMSprop می‌باشد یعنی از EMA استفاده می‌کند تنها تفاوتش در Epoch های اولیه می‌باشد زیرا در این روش از bias correction که در سوال ۲ توضیح دادم استفاده می‌کنیم این امر باعث می‌شود در epoch های اولیه طول حرکت زیادی نداشته باشیم و smooth تر حرکت کنیم. در epoch های جلوتر و در ادامه این روش همانند روش RMSprop رفتار می‌کند و ضابطه‌ی آنها با هم برابر می‌شود (در سوال ۲ اثبات شد)

Adam: از ترکیب تمامی روش‌های بالا به دست می‌آید. از Momentum به عنوان ممان اول، از RMSprop به عنوان ممان دوم استفاده می‌کند. همچنین bias correction که در روش AdaDelta نیز داشتیم را نیز بر روی هر دو ممان اعمال می‌کند. در نتیجه بهترین عملکرد را برای ما خواهد داشت.

مزایای دیگر Adam به صورت زیر است:

- پیاده‌سازی راحت
- سربار محاسباتی کم
- نیاز به حافظه زیادی ندارد
- هایپرپارامترهای آن مقدار مشخصی دارند و نیاز به tuning نیست (به جز learning rate)

ب)

به نوع دیتاست بستگی دارد برای مثال Adam بهترین عملکرد را در دیتاهای با Noise بالا و گرادیان Sparse دارد.

همچنین هر کدام از این بهینه‌سازها دارای هایپرپارامترهایی هستند که باید Fine Tune شوند و عملکرد شبکه بستگی به آن دارد.

اینکه بگوییم همیشه Adam نتیجه بهتری می‌دهد اشتباه است زیرا مقالات State of the art بسیاری را می‌بینیم که از گونه‌های SGD استفاده کرده‌اند.

همچنین در مقاله‌ای که لینک آن در ادامه آورده شده یک روش Hybrid پیشنهاد شده که ابتدا با Adam آموزش را شروع کنیم سپس هنگامی که یک شرطی رخ داد آن را به SGD تغییر دهیم. این کار باعث می‌شود که Generalization بهتری داشته باشیم.

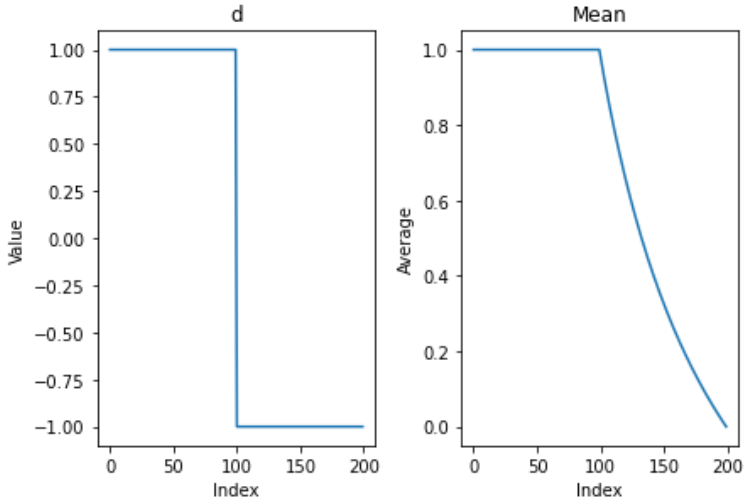
یکی دیگر از عوامل تاثیرگذار متریک و هدف ما می‌باشد. برای مثال اگر Convergence برای ما اهمیت بیشتری دارد بهتر است از Adam استفاده کنیم. اما اگر هدف Generalization است بهتر است از SGDNesterov استفاده کنیم. (مطالعه مقاله)

https://towardsdatascience.com/deep-learning-optimizers-436171c9e23f

https://arxiv.org/abs/1712.07628

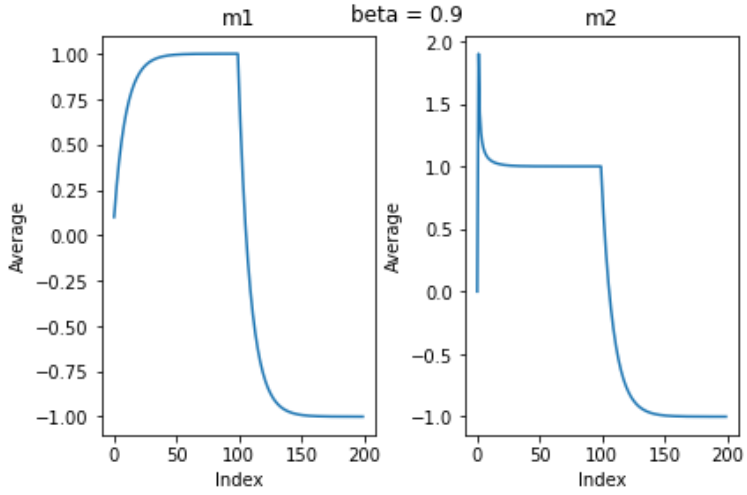
سوال ۲)

الف)



تا اندیس ۱۰۰ همه‌ی مقادیر برابر ۱ می‌باشند پس مقدار میانگین نیز ثابت و برابر ۱ خواهد بود. بعد از آن زمان با توجه به اینکه مقادیر d برابر ۱- می‌باشند میانگین به صورت اکیدا نزولی شروع به کاهش می‌کند.

ب - ج)



در واقع m1 همان Exponential moving average یا EMA می‌باشد و نحوه عملکرد آن به این صورت است که با توجه به مقدار beta به رکوردهای جدیدتر وزن نمایی بیشتری می‌دهد. در واقع مقدار کنونی آن به صورت زیر محاسبه می‌شود:

$$\mathrm{EMA}_{\mathrm{today}} = \frac{p_1 + (1-\alpha)p_2 + (1-\alpha)^2 p_3 + (1-\alpha)^3 p_4 + \cdots}{1 + (1-\alpha) + (1-\alpha)^2 + (1-\alpha)^3 + \cdots},$$

در این روش با توجه به اینکه نقطه جدید وزن 0.1 را دارد پس اثر آن به قدری خواهد بود تا بتواند اثر نقاط قبلی را جبران کند چون نقاط آخر d برابر -۱ می‌باشند پس سرعت کاهش زیاد خواهد و میانگین به سرعت همگرا می‌شود.

می‌توان مقدار EMA را با توجه به beta به صورت میانگین n نقطه آخر تقریب زد. که n به صورت زیر محاسبه می‌شود:

$$n = \frac{1}{1-\beta} = \frac{1}{1-0.9} = 10$$

پس می‌توان گفت مقدار *m1* تخمینی از مقدار میانگین در ۱۰ روز آخر می‌باشد.

m2 در واقع حالت *bias corrected* برای *m1* است. هنگامی که *t* مقادیر کمی دارد (نقاط اولیه) مقدار *m1* وابستگی بسیار زیادی به عبارت زیر دارد:

$$if\ t\ is\ little: \beta \times m_1[t-1] \approx 0$$
$$m_1[t] \propto (1-\beta) \times d[t]$$

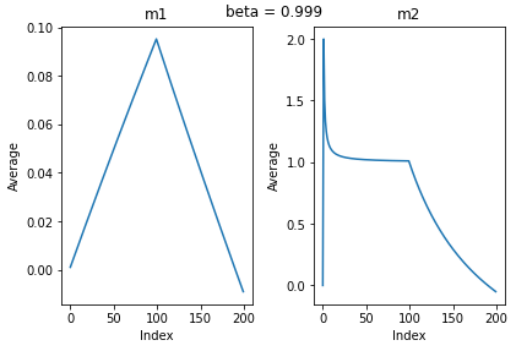
با توجه به اینکه مقدار $1-\beta$ کوچک است پس مقدار *m1* در ابتدا بسیار اختلاف زیادی با مقدار معنادار خواهد داشت. برای حل این مشکل حاصل *m1* را در عبارتی ضرب می‌کنیم تا این *bias* اصلاح شود:

$$m_2[t] = \frac{m_1[t]}{(1-\beta^t)}$$
$$if\ t\ is\ little \Longrightarrow (1-\beta^t) \approx 0 \Longrightarrow m_2[t]\ is\ high$$
$$if\ t\ is\ high \Longrightarrow (1-\beta^t) \approx 1 \Longrightarrow m_2[t] \approx\ m_1[t]$$

همانطور که در نمودار ها دیده می‌شود مقدار *m2* در ابتدا مقدار و شیب بالایی دارد و سپس همانند مقدار *d* می‌شود.

نمودار *m1* در ابتدا دارای *bias* زیادی است و در نقاط ابتدایی با شکل *d* کمی متفاوت است.

د)



در این حالت وزن نقاط جدید برابر *0.001* می‌باشد و ارزشش نسبت به حالت قبل کمتر است. پس الگوریتم دیرتر می‌تواند اثر نقاط مثبت قبلی را خنثی کند در نتیجه میانگین همگرا نمی‌شود.

مقدار n که بیانگر تقریبی از میانگین n روز آخر بود را محاسبه می‌کنیم:

$$n = \frac{1}{1-\beta} = \frac{1}{1-0.999} = 1000$$

در واقع مقدار *m1* برابر میانگین d در ۱۰۰۰ روز اخیر می‌باشد اما به دلیل بالا بودن *bias* در نقاط ابتدایی نمودار آن مشابه نمودار *Mean* در قسمت اول نمی‌شود.

اگر همانند سوال قبل این *bias* را اصلاح کنیم نمودار *m2* دقیقا شبیه نمودار *Mean* در قسمت اول می‌شود (به جز نقطه $t=0$)

دلیل این اتفاق این است که میانگین ۱۰۰۰ روز آخر در مسئله‌ای که کلا ۲۰۰ روز داریم برابر میانگین تجمعی می‌باشد. (۲۰۰ < ۱۰۰۰) بنابراین نمودار *m2* باید مطابق نمودار *Mean* شود.

https://en.wikipedia.org/wiki/Moving_average

سوال ۳)

الف) این دیتاست شامل تصاویر grayscale با ابعاد ۲۸ پیکسل در ۲۸ پیکسل می‌باشد که شامل ۶۰۰۰۰ داده آموزشی و ۱۰۰۰۰ داده آزمون است. هر یک از تصاویر این دیتاست یکی از ۱۰ گونه مربوط به پوشاک (Fashion) می‌باشد. این ۱۰ کلاس عبارتند از:

شکل تنسورهای این دیتاست نیز به صورت زیر است:

```
x_train:    (60000, 28, 28)
y_train:    (60000,)
x_test:     (10000, 28, 28)
y_test:     (10000,)
```

https://www.tensorflow.org/api_docs/python/tf/keras/datasets/fashion_mnist/load_data

https://www.researchgate.net/figure/Sample-images-from-Fashion-MNIST-dataset_fig2_342801790

ب) با توجه به اینکه دیتاست اصلی دارای ۶۰۰۰۰ داده آموزشی و ۱۰۰۰۰ داده آزمون است (نسبت ۶ به ۱) مقدار بین 0.16 تا 0.25 برای Validation Split می‌تواند مناسب باشد. در ابتدا مقدار 0.2 را انتخاب کردیم چون توزیع قبلی داده آموزش و تست را خیلی بر هم نمی‌زند. هم‌چنین داده‌های آموزشی به ۴۸۰۰۰ می‌رسد که همچنان مقدار مناسبی برای Generalize کردن شبکه می‌باشد.

```
Number of train examples:          48000
Number of validation examples:     12000
Number of test examples:           10000
Train / Validation:                4
```

- ✓ در قسمت (د) این مقدار را ۳ بار تغییر می‌دهیم.
- ✓ همانطور که می‌دانید تابع train_test_split در ابزار sklearn پیش‌فرض این مقدار را 0.25 در نظر می‌گیرد. (عدم استفاده)

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

ج) شبکه را با هایپرپارامترهای مشترک زیر آموزش و ارزیابی می‌کنیم:

```
BATCH_SIZE = 64
LOSS = "sparse_categorical_crossentropy"
LEARNING_RATE = 0.001
OPTIMIZER = SGD(learning_rate=LEARNING_RATE)
EPOCHS = 50
VAL_SPLIT = 0.2
```

نتایج برای تعداد نورون‌های مختلف لایه مخفی به‌صورت زیر است:

Hidden Units	Train	Validation	Test
16	loss: 1.2465 accuracy: 0.4728	loss: 1.3133 accuracy: 0.4586	loss: 1.3189 accuracy: 0.4650
32	loss: 0.6558 accuracy: 0.7656	loss: 0.7129 accuracy: 0.7642	loss: 0.7478 accuracy: 0.7738
64	loss: 0.4820 accuracy: 0.8207	loss: 0.5902 accuracy: 0.7912	loss: 0.6263 accuracy: 0.7862
128	loss: 0.3869 accuracy: 0.8595	loss: 0.5678 accuracy: 0.8308	loss: 0.5752 accuracy: 0.8196

✓ نتایج به‌صورت کامل در نوتبوک موجود است (به ازای هر Epoch)

تحلیل: مقدار loss و accuracy برای تعداد نورون‌های مختلف به صورت زیر مقایسه میشود:

loss: 16 > 32 > 64 > 128

accuracy: 128> 64 > 32 > 16

پس می‌توان نتیجه گرفت برای دیتاست بالا تعداد ۱۲۸ نورون در لایه مخفی بهترین حالت است.

هم‌چنین اگر مقدار loss و accuracy را برای ۳ بخش Train و Validation و Test مقایسه کنیم داریم:

- 16 Hidden Units

loss: Test > Validation > Train

accuracy: Train = 47% > Test > Validation

حتی در حالت آموزش دقت و خطا مناسبی نداشتیم و شبکه دارای high bias بوده و underfit شده. پس اندازه شبکه یعنی ۱۶ نورون مناسب نبوده.

- 32 Hidden Units

loss: Test > Validation > Train

accuracy: Test = 77% > Train > Validation

نسبت به حالت قبلی مشکل high bias و underfit کمی حل شده اما همچنان دقت و خطای فاز آموزش مناسب نیست و باید شبکه را بزرگتر کرد.

- 64 Hidden Units

loss: Test > Validation > Train

accuracy: Train = 82% > Validation > Test

نسبت به حالات قبلی مشکل high bias و underfit کاملا حل شده و دیگر مشکل حالت قبلی که دقت آزمون بیشتر از آموزش بود را ندارد. اما هم‌چنان می‌توانیم با بزرگ کردن شبکه دقت را بیشتر کنیم و bias را کاهش دهیم.

- 128 Hidden Units

loss: Test > Validation > Train

accuracy: Train = 86% > Validation > Test

از نظر دقت و خطا بهترین شبکه (در این مسئله) می‌باشد. نسبت به حالت ۶۴ نورون هم خطا کاهش یافته (چشمگیر) هم دقت افزایش.

- ✓ در هیچکدام از حالات مشکل high variance و overfit پیش‌نیامد زیرا باقی هایپرپارامتر ها مناسب انتخاب شده بودند.
- ✓ خطا در داده آزمون از داده‌های دیگر بیشتر است زیرا تعداد ورودی‌های آن از داده ارزیابی کمتر است و بر خلاف داده آموزش هیچ آموزشی روی آن صورت نمی‌گیرد.
- ✓ نمودارهای Loss و Accuracy به ازای Epoch در قسمت (ح) رسم شده‌است.

د) مقدار Validation Split را ۳ بار به مقادیر زیر تغییر می‌دهیم و به ازای تعداد نورون‌هایی که در صورت سوال بود نتیجه را بررسی می‌کنیم. هایپرپارامترهای مشترک شبکه به‌صورت زیر است:

```
BATCH_SIZE = 64
LOSS = "sparse_categorical_crossentropy"
LEARNING_RATE = 0.001
OPTIMIZER = SGD(learning_rate=LEARNING_RATE)
EPOCHS = 50
```

VAL_SPLIT = 0.15

Hidden Units	Train	Validation	Test
16	loss: 1.2444 accuracy: 0.4986	loss: 1.2937 accuracy: 0.4950	loss: 1.2797 accuracy: 0.5070
32	loss: 0.6294 accuracy: 0.7645	loss: 0.6947 accuracy: 0.7409	loss: 0.6954 accuracy: 0.7485
64	loss: 0.5302 accuracy: 0.7986	loss: 0.6776 accuracy: 0.7664	loss: 0.7026 accuracy: 0.7664
128	loss: 0.3809 accuracy: 0.8617	loss: 0.5509 accuracy: 0.8280	loss: 0.5731 accuracy: 0.8251

VAL_SPLIT = 0.20 Calculated in previous parts

Hidden Units	Train	Validation	Test
16	loss: 1.2465 accuracy: 0.4728	loss: 1.3133 accuracy: 0.4586	loss: 1.3189 accuracy: 0.4650
32	loss: 0.6558 accuracy: 0.7656	loss: 0.7129 accuracy: 0.7642	loss: 0.7478 accuracy: 0.7738
64	loss: 0.4820 accuracy: 0.8207	loss: 0.5902 accuracy: 0.7912	loss: 0.6263 accuracy: 0.7862
128	loss: 0.3869 accuracy: 0.8595	loss: 0.5678 accuracy: 0.8308	loss: 0.5752 accuracy: 0.8196

VAL_SPLIT = 0.25

Hidden Units	Train	Validation	Test
16	loss: 1.5996 accuracy: 0.3580	loss: 1.6443 accuracy: 0.3449	loss: 1.6420 accuracy: 0.3503
32	loss: 0.8174 accuracy: 0.6979	loss: 1.2648 accuracy: 0.6074	loss: 1.2805 accuracy: 0.6019
64	loss: 0.4617 accuracy: 0.8288	loss: 0.5537 accuracy: 0.8184	loss: 0.5700 accuracy: 0.8168
128	loss: 0.3986 accuracy: 0.8557	loss: 0.5456 accuracy: 0.8302	loss: 0.5671 accuracy: 0.8242

VAL_SPLIT = 0.30

Hidden Units	Train	Validation	Test
16	loss: 1.3409 accuracy: 0.4479	loss: 1.5924 accuracy: 0.3762	loss: 1.5850 accuracy: 0.3747
32	loss: 0.6611 accuracy: 0.7348	loss: 0.7440 accuracy: 0.7302	loss: 0.7599 accuracy: 0.7230
64	loss: 0.6592 accuracy: 0.7478	loss: 0.8403 accuracy: 0.7126	loss: 0.8245 accuracy: 0.7096
128	loss: 0.3848 accuracy: 0.8579	loss: 0.5337 accuracy: 0.8354	loss: 0.5546 accuracy: 0.8252

تحلیل: در هر ۴ حالتی که برای Validation Split داشتیم مقدار loss و accuracy برای تعداد نورون‌های مختلف به صورت زیر مقایسه میشود:

loss: 16 > 32 > 64 > 128

accuracy: 128> 64 > 32 > 16

- پس در انتخاب بهترین تعداد نورون‌های لایه مخفی تغییری ایجاد نشد. در واقع هایپرپارامتر Validation Split بر سایر هایپرپارامترها عمود است (Orthogonality) و تعداد نورون لایه مخفی به آن وابسته نیست و می‌توان بهترین مقدار آن را مستقل از Validation Split به دست آورد.
- دقت فاز آزمون برای تعداد نورون‌های مختلف و Validation Split های مختلف به صورت زیر مقایسه می‌شود:

- 16 Hidden Units

test accuracy: VS (0.15) > VS (0.20) > VS (0.30) >= VS (0.25)

test loss: VS (0.15) > VS (0.20) > VS (0.30) > VS (0.25)

- 32 Hidden Units

test accuracy: VS (0.20) > VS (0.15) > VS (0.30) > VS (0.25)

test loss: VS (0.15) > VS (0.20) > VS (0.30) > VS (0.25)

- 64 Hidden Units

test accuracy: VS (0.25) > VS (0.20) > VS (0.15) >= VS (0.30)

test loss: VS (0.25) > VS (0.20) > VS (0.15) > VS (0.30)

- 128 Hidden Units

test accuracy: VS (0.30) >== VS (0.15) >= VS (0.25) >= VS (0.20)

test loss: VS (0.30) > VS (0.25) > VS (0.15) > VS (0.20)

ترتیب و رابطه‌ی مشخصی دیده نمی‌شود اما نکات زیر قابل توجه است:

۱- در حالت ۱۲۸ نورون نتایج بسیار نزدیک هستند و Validation Split تاثیر بسیار کمی دارد.

۲- دقت داده تست در حالت ۶۴ نورون با VS = 0.25 مقدار بسیار خوبی دارد (بسیار نزدیک به حالت ۱۲۸ نورون) پس می‌توان با انتخاب آن در این شبکه کوچک‌تر نتیجه مطلوب گرفت. همین موضوع باعث می‌شود مقدار Validation Split را برابر 0.25 در نظر بگیریم. دقیقا همان مقداری که کتاب‌خانه sklearn در تابع خودش به صورت پیش‌فرض در نظر گرفته.

تاثیر این پارامتر به این گونه است که اگر مقدار آن کم باشد اکثر داده‌ها به بخش آموزش منتقل می‌شود و شبکه آموزش بهتری می‌بیند ما با ضعیف بودن ارزیابی دقت داده آزمون کاهش می‌یابد.

اگر این مقدار بیش از اندازه بزرگ باشه شبکه داده آموزشی کمتری می‌بیند و ممکن است روی حجم داده آموزشی کم Overfit کند و نتواند روی داده آزمون به خوبی Generalize کند.

ه) شبکه را با هایپرپارامترهای مشترک زیر و تنها با تغییر بهینه‌ساز آموزش و ارزیابی می کنیم.

```
VAL_SPLIT = 0.25
HIDDEN_UNITS = 128
LEARNING_RATE = 0.001
EPOCHS = 50
```

Optimizer	Train	Validation	Test
SGD	loss: 0.3986 accuracy: 0.8557	loss: 0.5456 accuracy: 0.8302	loss: 0.5671 accuracy: 0.8242
Adam	loss: 0.3469 accuracy: 0.8778	loss: 0.5567 accuracy: 0.8514	loss: 0.5576 accuracy: 0.8457
RMSprop	loss: 0.4952 accuracy: 0.8653	loss: 1.6664 accuracy: 0.8326	loss: 1.7180 accuracy: 0.8248
Adagrad	loss: 0.5100 accuracy: 0.8640	loss: 1.7209 accuracy: 0.8315	loss: 1.8189 accuracy: 0.8208

مقایسه:

Train loss: Adam < SGD < RMSprop < Adagrad

Val loss: SGD < Adam < RMSprop < Adagrad

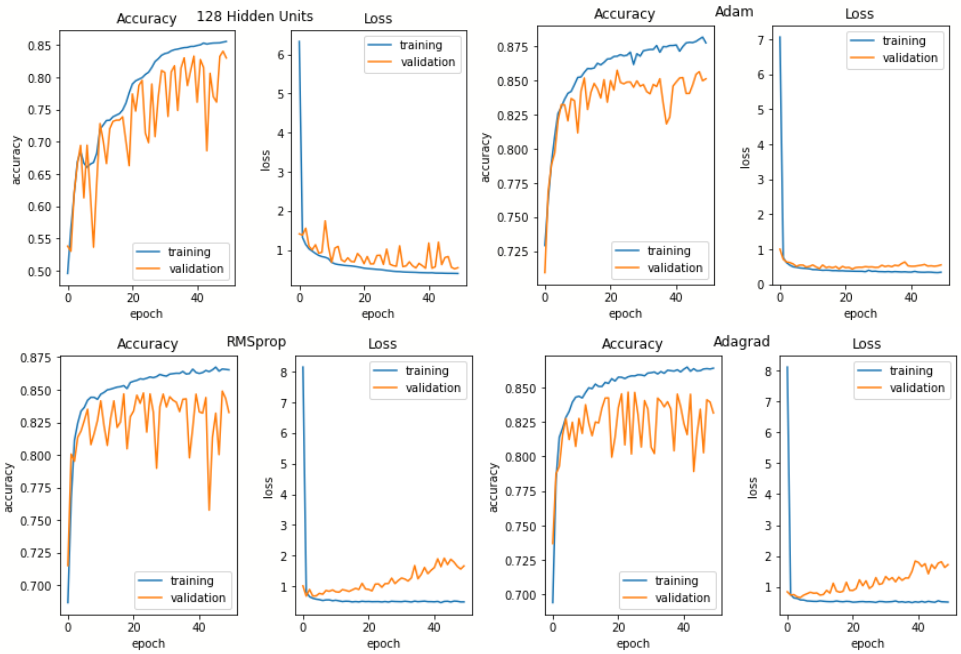
Test loss: Adam < SGD < RMSprop < Adagrad

Train accuracy: Adam > RMSprop > Adagrad > SGD

Val accuracy: Adam > RMSprop > Adagrad > SGD

Test accuracy: Adam > RMSprop > SGD > Adagrad

Adam بهترین عملکرد را دارد. زیرا ترکیبی از تمامی حالت هاست و امکان دور افتادن از Global Optimum در آن کم است. همچنین نوسان در آن کمتر می‌باشد. نوسان زیاد را می‌توان در نمودار سایر بهینه‌سازها مشاهده کرد.



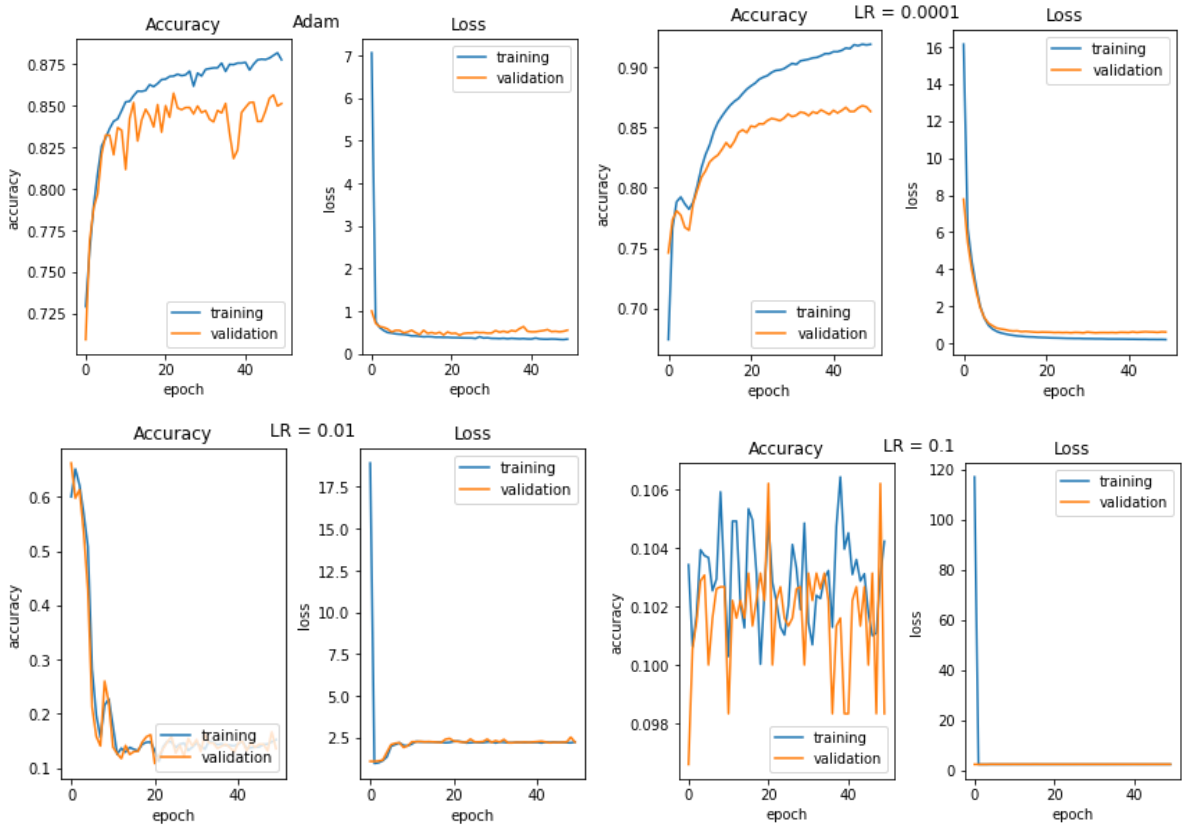
و) شبکه را با هایپرپارامترهای مشترک زیر و تنها با تغییر نرخ آموزش، آموزش و ارزیابی می‌کنیم.

```
VAL_SPLIT = 0.25
HIDDEN_UNITS = 128
OPTIMIZER = Adam(learning_rate=LEARNING_RATE)
EPOCHS = 50
```

Learning Rate	Train	Validation	Test
0.001	loss: 0.3544 accuracy: 0.8736	loss: 0.5554 accuracy: 0.8467	loss: 0.5788 accuracy: 0.8383
0.0001	loss: 0.2129 accuracy: 0.9187	loss: 0.6215 accuracy: 0.8632	loss: 0.6339 accuracy: 0.8589
0.01	loss: 2.2262 accuracy: 0.1521	loss: 2.2202 accuracy: 0.1356	loss: 2.2004 accuracy: 0.1412
0.1	loss: 2.3048 accuracy: 0.1042	loss: 2.3684 accuracy: 0.0983	loss: 2.2972 accuracy: 0.1037

در حالت 0.1 و 0.01 شبکه دارای bias بسیار زیاد می‌باشد و عملا fit نمی‌شود زیرا این مقدار قدم‌های گرادیان کاهشی را بسیار بزرگ در نظر می‌گیرد و باعث می‌شود مدام از نقطه بهینه بپریم. عدم همگرایی

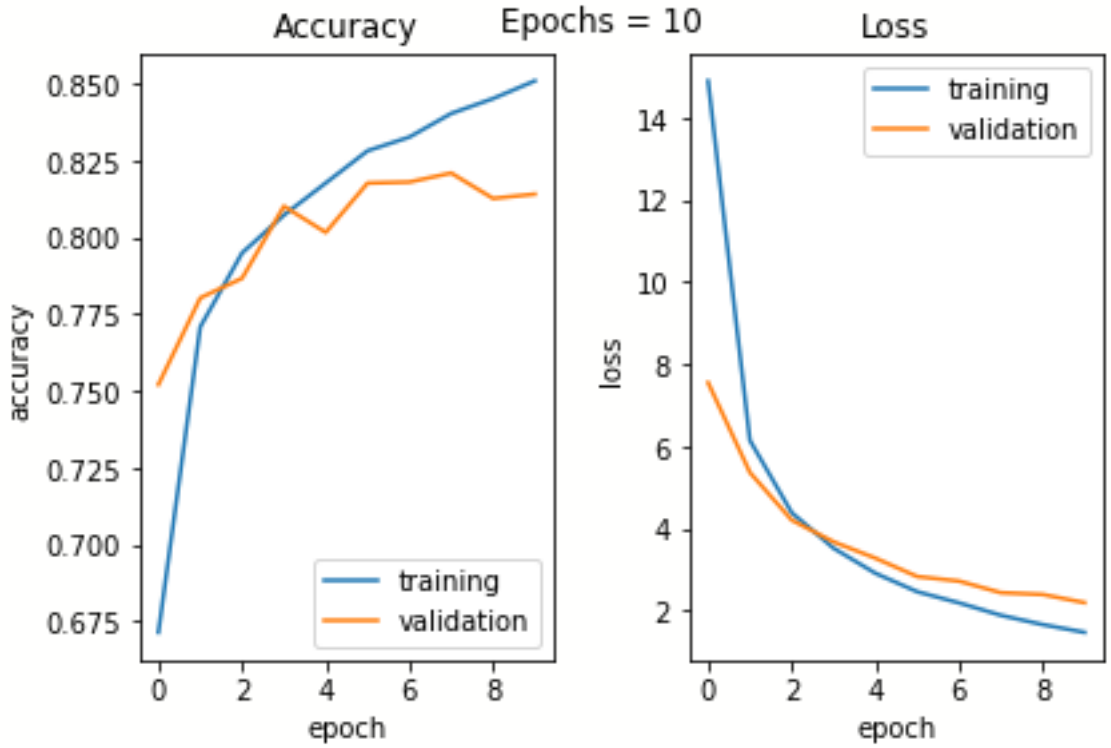
حالت 0.001 نسبت به 0.0001 دارای Loss کمتری است. Accuracy نیز کمتر می‌باشد (نوسان زیاد). پس انتخاب بهترین نرخ آموزش بستگی به Metric ما دارد که در صورت سوال دقت (Accuracy) مورد پرسش است پس مقدار 0.0001 را انتخاب می‌کنیم.



ز) شبکه را با هایپرپارامترهای زیر و Epoch برابر ۱۰ آموزش و ارزشیابی می کنیم.

```
VAL_SPLIT = 0.25
HIDDEN_UNITS = 128
LEARNING_RATE = 0.0001
OPTIMIZER = Adam(learning_rate=LEARNING_RATE)
```

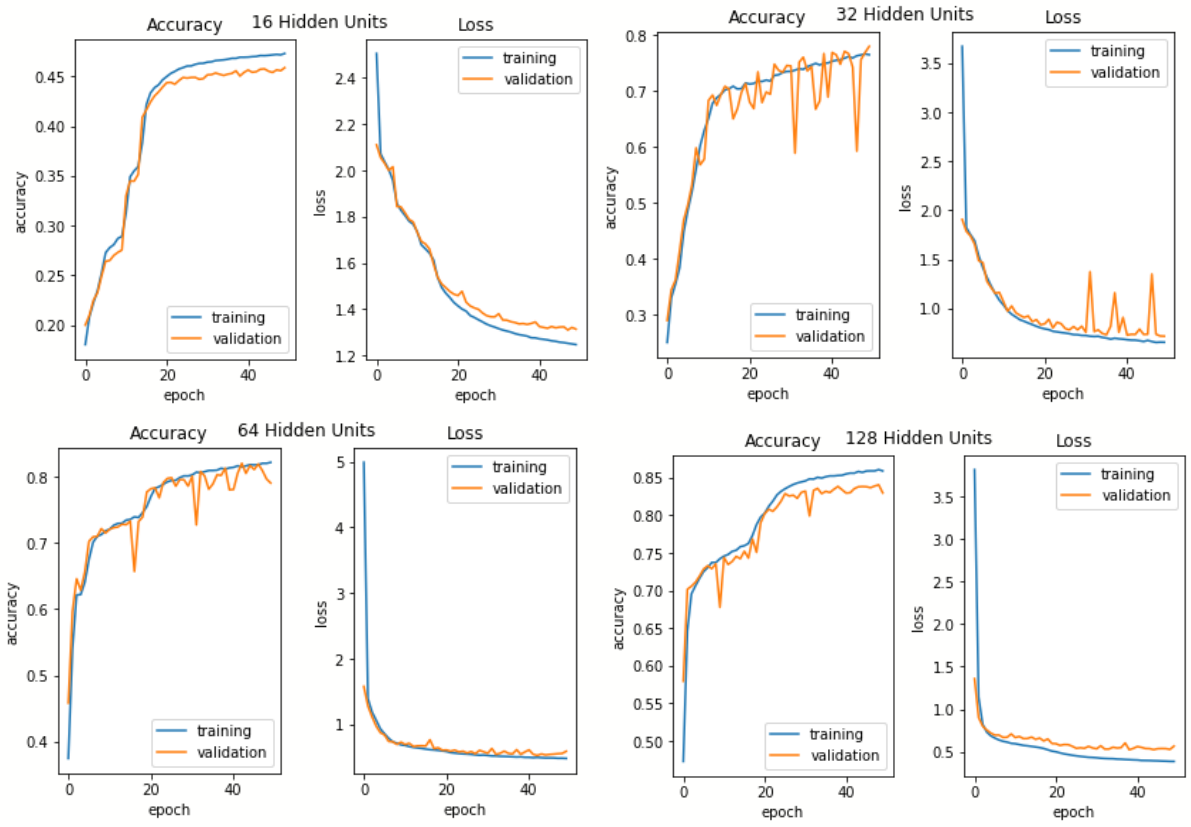
Epochs	Train	Validation	Test
50	loss: 0.2129 accuracy: 0.9187	loss: 0.6215 accuracy: 0.8632	loss: 0.6339 accuracy: 0.8589
10	loss: 1.4732 accuracy: 0.8510	loss: 2.1958 accuracy: 0.8141	loss: 2.3087 accuracy: 0.8077



با توجه به نمودار Loss می‌توان گفت شبکه به طور کامل همگرا نشده. از مقایسه نتایج داخل جدول نیز می‌توان به این نکته پی برد زیرا مقدار Loss بزرگ است.

ح) مشابه قسمت (ج)

```
BATCH_SIZE = 64
LOSS = "sparse_categorical_crossentropy"
LEARNING_RATE = 0.001
OPTIMIZER = SGD(learning_rate=LEARNING_RATE)
EPOCHS = 50
VAL_SPLIT = 0.2
```



۱۶ نورون: حتی در حالت آموزش دقت و خطا مناسبی نداشتیم و شبکه دارای high bias بوده و underfit شده. پس اندازه شبکه یعنی ۱۶ نورون مناسب نبوده. (شبکه بسیار سریع همگرا شده)

۳۲ نورون: نسبت به حالت قبلی مشکل high bias و underfit کمی حل شده اما همچنان دقت و خطای فاز آموزش مناسب نیست و باید شبکه را بزرگتر کرد. همچنین نوسان و پرش در این حالت بسیار زیاد است و احتمالا مشکل high variance هم داریم.

۶۴ نورون: نسبت به حالات قبلی مشکل high bias و underfit کاملا حل شده و دیگر مشکل حالت قبلی را ندارد. اما هم‌چنان می‌توانیم با بزرگ کردن شبکه دقت را بیشتر کنیم و bias را کاهش دهیم. در این حالت همگرایی به خوبی رخ داده و نوسانات نیز کم است.

۱۲۸ نورون: از نظر دقت و خطا بهترین شبکه (در این مسئله) می‌باشد. نسبت به حالت ۶۴ نورون هم خطا کاهش یافته (چشمگیر) هم دقت افزایش. همچنین همگرایی حفظ شده و نوسانات نیز کم می‌باشد.