به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

# درس شبکه‌های عصبی و یادگیری عمیق

## تمرین اضافه

| | |
|---|---|
| نام و نام خانوادگی | امیرحسین پورداود |
| شماره دانشجویی | ۸۱۰۱۰۱۱۲۰ |
| تاریخ ارسال گزارش | ۱۴۰۱.۰۹.۲۸ |

# فهرست

# شکل‌ها

# جدول‌ها

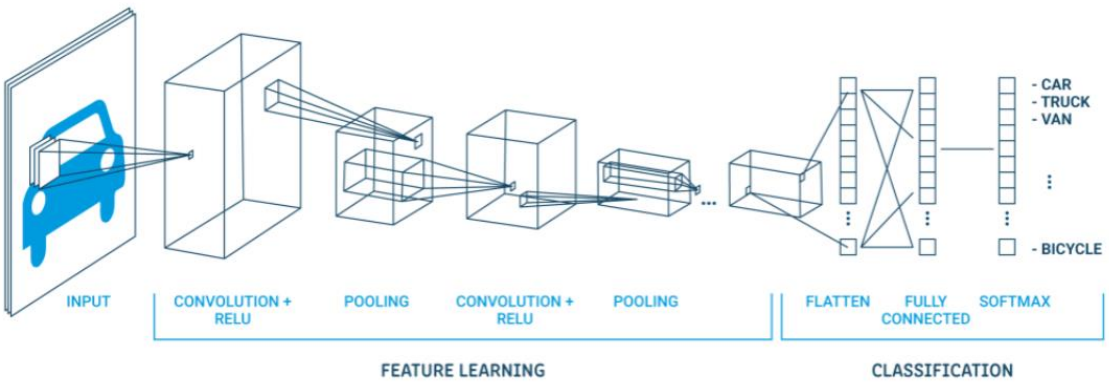
# پاسخ ۳ – تشخیص کاراکتر نوری (Opical character recognition)

## ۳-۱. تفاوت بین شبکه های CNN و DCNN

تفاوت بین CNN و Deep CNN فقط در تعداد لایه ها است. یک مدل یادگیری عمیق زمانی عمیق تر نامیده می شود که تعداد لایه های بیشتری داشته باشید. بنابراین، Deep CNN اساساً CNN با لایه های عمیق تر است. در CNN معمولی، معمولاً ۵ تا ۱۰ لایه وجود دارد، در حالی که بیشتر معماری های CNN مدرن ۳۰ تا ۱۰۰ لایه عمیق دارند که برای استخراج ویژگی های بیشتر و افزایش دقت پیش بینی استفاده می شود. دو نوع CNN عمیق وجود دارد، یکی افزایش تعداد لایه های پنهان یا با افزایش تعداد گره ها در لایه پنهان ایجاد میشود.

یک DCNN از یک شبکه عصبی سه بعدی برای پردازش عناصر قرمز، سبز و آبی تصویر به طور همزمان استفاده می کند. این به طور قابل توجهی تعداد نورون های مصنوعی مورد نیاز برای پردازش یک تصویر را در مقایسه با شبکه های عصبی پیشخور سنتی کاهش می دهد. شبکه های عصبی کانولوشنال عمیق تصاویر را به عنوان ورودی دریافت می کنند و از آنها برای آموزش یک طبقه بندی کننده استفاده می کنند. این شبکه به جای ضرب ماتریس از یک عملیات ریاضی خاص به نام "پیچیدگی" استفاده می کند.

معماری یک شبکه کانولوشن معمولا از چهار نوع لایه تشکیل شده است: کانولوشن، ادغام، فعال سازی و کاملا متصل.




شکل ۱ نمونه ای از **DCNN**

## ۳-۲. روش بهینه سازی Adam و Adadelta و Momentum

### ۳-۲-۱- *Adam*

Adam مزایای دو نمونه دیگر از نزول گرادیان تصادفی را ترکیب کرده است. به طور مشخص:

- Adaptive Gradient Algorithm (AdaGrad) که نرخ یادگیری در هر پارامتر را حفظ می‌کند و عملکرد را در مشکلات با گرادیان‌های پراکنده (مانند مشکلات زبان طبیعی و بینایی ماشین) بهبود می‌بخشد.
- Root Mean Square Propagation (RMSProp) که همچنین نرخ‌های یادگیری هر پارامتر را حفظ می‌کند و بر اساس میانگین بزرگی‌های اخیر گرادیان‌ها برای وزن (مثلاً سرعت تغییر آن) تنظیم می‌شود. این بدان معناست که الگوریتم در مسائل آنلاین و غیر ثابت (به عنوان مثال نویز) به خوبی عمل می کند.

آدام از مزایای AdaGrad و RMSProp بهره می برد. با این حال، به طور تصاعدی گرادیان مرتبه اول را هموار می کند تا momentum را در به روز رسانی بگنجاند. همچنین به طور مستقیم به انحراف ذاتی در هموارسازی نمایی می پردازد که تخمین در حال اجرا یک مقدار هموار شده به طور غیر واقعی به ۰ مقداردهی اولیه شود.

Adam یک روش نرخ یادگیری تطبیقی است، به این معنی که نرخ یادگیری فردی را برای پارامترهای مختلف محاسبه می کند. نام آن از adaptive moment estimation گرفته شده است و دلیل نامگذاری آن به این دلیل است که آدام از تخمین لحظه های اول و دوم گرادیان برای تطبیق نرخ یادگیری برای هر وزن شبکه عصبی استفاده می کند. در Adam به جای تطبیق نرخ یادگیری بر اساس میانگین لحظه اول مانند RMSP، Adam از میانگین لحظات دوم گرادیان ها استفاده می کند. آدم. این الگوریتم میانگین متحرک نمایی گرادیان و گرادیان مربع را محاسبه می کند. و از پارامترهای β1 و β2 برای کنترل نرخ پوسیدگی این میانگین متحرک استفاده می شود.

باتوجه به RMSProp، که $A_i$ مقدار میانگین نمایی iامین پارامتر $w_i$ است. این مقدار به همان روش RMSProp با پارامتر (decay) واپاشی $\rho \in (0, 1)$ به روز می شود:

$$A_i \Leftarrow \rho A_i + (1-\rho)\left(\frac{\partial L}{\partial w_i}\right)^2 \quad \forall i$$

در همان زمان، یک مقدار هموار شده نمایی از گرادیان حفظ می شود که مولفه i برای آن با Fi نشان داده می شود. این هموارسازی با پارامتر واپاشی متفاوتی $\rho_f$ انجام می شود:

$$F_i \Leftarrow \rho_f F_i + (1 - \rho_f) \left( \frac{\partial L}{\partial w_i} \right) \quad \forall i$$

این نوع هموارسازی نمایی گرادیان با $\rho_f$ تغییری از روش momentum است (که با پارامتر اصطکاک $\beta$ به جای $\rho_f$ پارامتر می شود). سپس، به روز رسانی زیر با نرخ یادگیری $\alpha_t$ در تکرار t استفاده می شود:

$$w_i \Leftarrow w_i - \frac{\alpha_t}{\sqrt{A_i}} F_i; \quad \forall i$$

دو تفاوت کلیدی با الگوریتم RMSProp وجود دارد. ابتدا، گرادیان با مقدار هموار شده آن به صورت نمایی جایگزین می شود تا تکانه را در خود جای دهد. دوم، نرخ یادگیری $\alpha_t$ در حال حاضر به شاخص تکرار t بستگی دارد و به صورت زیر تعریف می شود:

$$\alpha_t = \alpha \underbrace{\left( \frac{\sqrt{1 - \rho^t}}{1 - \rho_f^t} \right)}_{\text{Adjust Bias}}$$

از نظر فنی، تنظیم نرخ یادگیری در واقع یک عامل تصحیح سوگیری است که برای مقدار اولیه‌سازی غیرواقعی دو مکانیسم هموارسازی نمایی اعمال می‌شود و به ویژه در تکرارهای اولیه اهمیت دارد.

### ۳-۲-۲- *Adadelta*

الگوریتم AdaDelta از یک به روز رسانی مشابه به عنوان RMSProp استفاده می کند، با این تفاوت که با محاسبه آن به عنوان تابعی از به روز رسانی های افزایشی در تکرارهای قبلی، نیاز به پارامتر یادگیری سراسری را از بین می برد. به روز رسانی RMSProp را در نظر بگیرید که در زیر تکرار می شود:

$$w_i \Leftarrow w_i - \underbrace{\frac{\alpha}{\sqrt{A_i}} \left( \frac{\partial L}{\partial w_i} \right)}_{\Delta w_i}; \quad \forall i$$

نشان داده شده است که چگونه α با مقداری که به بروز رسانی های افزایشی قبلی بستگی دارد جایگزین می شود. در هر به روز رسانی، wi به مقدار Δwi افزایش می یابد. همانند گرادیان های نرم شده نمایی Ai، در اینجا یک مقدار هموار شده نمایی δi از مقادیر Δwi را در تکرارهای قبلی با همان پارامتر واپاشی ρ حفظ شده است:

$$\delta_i \Leftarrow \rho\delta_i + (1-\rho)(\Delta w_i)^2 \quad \forall i$$

برای یک تکرار معین، مقدار δi را می توان تنها با استفاده از تکرارهای قبل از آن محاسبه کرد زیرا مقدار Δwi هنوز در دسترس نیست. از سوی دیگر، Ai را می توان با استفاده از مشتق جزئی در تکرار جاری نیز محاسبه کرد. این یک تفاوت ظریف بین نحوه محاسبه Ai و δi است. این منجر به به‌روزرسانی AdaDelta زیر می‌شود:

$$w_i \Leftarrow w_i - \underbrace{\sqrt{\frac{\delta_i}{A_i}}\left(\frac{\partial L}{\partial w_i}\right)}_{\Delta w_i}; \quad \forall i$$

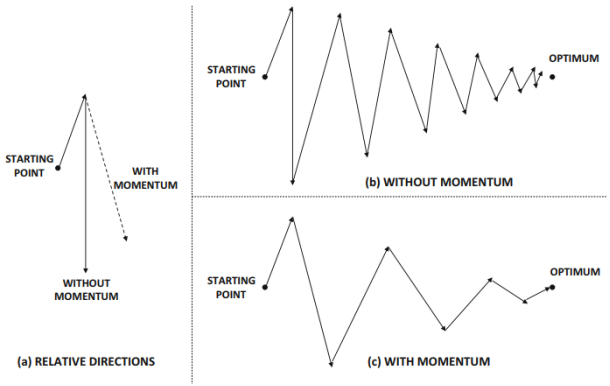
قابل ذکر است که پارامتر α برای نرخ یادگیری به طور کامل در این به روز رسانی وجود ندارد. روش AdaDelta شباهت هایی با روش های مرتبه دوم دارد زیرا نسبت sqrt(δi/Ai) در به روز رسانی یک جایگزین اکتشافی برای معکوس مشتق دوم loss با توجه به wi است. بسیاری از روش‌های مرتبه دوم مانند روش نیوتن نیز از نرخ یادگیری استفاده نمی‌کنند.

### ۳-۲-۳- *Momentum*

زمان صرف شده برای آموزش مدل بوسیله SGD نسبتاً بالاتر است. بنابراین، برای انجام همگرایی سریعتر، SGD با بهینه سازهای دیگر مانند adadelta، Adam و momentum بهینه می شود. وقتی در مورد تکانه صحبت می کنیم، توپ در حال حرکت به سمت پایین تپه به جمع آوری تکانه ادامه می دهد و در مسیر خود شتاب می گیرد تا به سرعت نهایی برسد.

تکنیک‌های مبتنی بر تکانه تشخیص می‌دهند که زیگزاگی نتیجه گام‌های بسیار متناقضی است که یکدیگر را خنثی می‌کنند و اندازه مؤثر مراحل را در جهت صحیح (دراز مدت) کاهش می‌دهند. نمونه ای از این سناریو در شکل زیر نشان داده شده است. صرفاً تلاش برای افزایش اندازه گام به منظور دستیابی به حرکت بیشتر در جهت صحیح ممکن است در واقع راه حل فعلی را حتی بیشتر

از راه حل بهینه دور کند. از این منظر، حرکت در جهت «متوسط» چند قدم آخر بسیار منطقی است، به طوری که زیگزاگ هموار شود.




شکل ۲ - تاثیر تکانه در جهت یابی سطوح

به منظور درک این نکته، مثالی را در نظر بگیرید که در آن یک گرادیان نزولی را با توجه به بردار پارامتر W انجام می‌دهد. به‌روزرسانی‌های عادی برای گرادیان نزولی با توجه به تابع L loss (تعریف شده در یک دسته کوچک از نمونه‌ها) به شرح زیر است:

$$\overline{V} \Leftarrow -\alpha \frac{\partial L}{\partial \overline{W}}; \quad \overline{W} \Leftarrow \overline{W} + \overline{V}$$

در اینجا α نرخ یادگیری است. در فرود مبتنی بر تکانه، بردار V با هموارسازی نمایی اصلاح می‌شود، که ∃β (۰، ۱) یک پارامتر هموارسازی است:

$$\overline{V} \Leftarrow \beta \overline{V} - \alpha \frac{\partial L}{\partial \overline{W}}; \quad \overline{W} \Leftarrow \overline{W} + \overline{V}$$

مقادیر بزرگتر β به رویکرد کمک می کند تا یک سرعت ثابت V را در جهت صحیح بگیرد. تنظیم β = 0 برای نزول گرادیان دسته ای ساده است. پارامتر β به عنوان پارامتر تکانه یا پارامتر اصطکاک نیز نامیده می شود. کلمه "اصطکاک" از این واقعیت گرفته شده است که مقادیر کوچک β به عنوان "ترمز" عمل می کنند، بسیار شبیه اصطکاک. با نزول مبتنی بر تکانه، یادگیری تسریع می‌شود، زیرا فرد معمولاً در جهتی حرکت می‌کند که اغلب به راه‌حل بهینه نزدیک‌تر می‌شود و نوسان‌های بی‌فایده «کنار» خاموش می‌شوند. ایده اصلی این است که اولویت بیشتری به جهت های منسجم نسبت به مراحل متعدد داده شود، که در فرود اهمیت بیشتری دارند. این امکان استفاده از پله های بزرگتر را در جهت صحیح بدون ایجاد سرریز یا "انفجار" در جهت جانبی فراهم می کند.

استفاده از تکانه اغلب باعث می‌شود که محلول در جهتی که سرعت گرفته می‌شود کمی بیش از حد شود، درست همانطور که یک تیله زمانی که اجازه می‌دهد روی یک کاسه غلت بخورد، بیش از حد بالا می‌رود. با این حال، با انتخاب مناسب β، باز هم عملکرد بهتری نسبت به موقعیتی دارد

که در آن از تکانه استفاده نمی شود. روش مبتنی بر تکانه عموماً عملکرد بهتری خواهد داشت زیرا تیله با غلتیدن روی کاسه سرعت می‌گیرد. رسیدن سریعتر به راه‌حل بهینه، بیش از آن که بیش از حد هدف را جبران کند. بیش از حد مطلوب است تا حدی که به جلوگیری از بهینه محلی کمک کند.

## ۳-۳. پیاده سازی مدل DCNN

معماری استفاده شده در این مقاله یک شبکه DCNN است که شامل ۳ بلوک کانولوشنی و چند لایه dense میباشد. هر یک از بلوک ها شامل لایه های کانولوشنی با تابع فعالساز Relu، لایه MaxPooling، لایه Dropout و لایه Batch Normalization میباشد که ترتیب آن ها بصورت جدول زیر آورده شده است و طبق این جدول پیاده سازی انجام گرفته است.

جدول ۱ - پارامتر های لایه های بکار رفته در معماری **DCNN**

| Block | No of Filter | Stride | Layers | Filter Size |
|---|---|---|---|---|
| Image Input | - | - | 40*40*1 | - |
| Feature Extraction | | | | |
| 1st Conv-Block | 64 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | - | 2*2 | Pooling Layer | 3*3 |
| | - | - | Dropout (0.1) | - |
| 2nd Conv-Block | 128 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | 128 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | - | 2*2 | Pooling Layer | 3*3 |
| | - | - | Dropout (0.2) | - |
| 3rd Conv-Block | 256 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | 256 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | - | 2*2 | Pooling Layer | 3*3 |
| | - | - | Dropout (0.3) | - |
| 4th Conv-Block | 512 | 1*1 | Conv Layer | 3*3 |
| | - | 1*1 | Activation Layer (Relu) | - |
| | - | - | Batch Normalization | - |
| | 512 | 1*1 | Conv Layer | 3*3 |
| | - | - | Batch Normalization | - |
| | - | 2*2 | Pooling Layer | 3*3 |
| | - | - | Dropout (0.4) | - |
| Classification Block | | | | |
| | - | - | Flatten Layer | - |
| | - | - | FC Layer (1024) | - |
| | - | - | Batch Normalization | - |
| | - | - | Dropout (0.5) | - |
| | - | - | FC Layer (10 classes) | - |
| | - | - | Output | - |

داده های مورد استفاده برای آموزش مدل از دیتاست HODA استفاده شده است که شامل تقریبا ۸۰۰۰۰ داده تست و آموزش میباشد. عکس های موجود در این دیتاست دارای سایز های مختلف میباشد که برای آموزش نیاز است که سایز تمامی عکس ها یکی شود. پیش پردازش ها میتواند شامل فیلتر های رفع نویز و روشن کردن قسمت های مهم و باینری کردن تصاویر و .. میباشد که در زیر پیش پردازش استفاده شده در مقاله را شرح میدهیم:

- تغییر سایز تصویر به ابعاد ۴۰*۴۰ برای آموزش شبکه که الگوی اصلی در وسط تصویر باقی بماند.
- تبدیل تصویر بصورت باینری برای آموزش بهتر مدل که ۰ و ۱ باشد و وزن ها و تشخیص بهتر عمل کند.
- تغییر پس زمینه تصویر به ۰ و رنگ مشکی و تصویر عدد به رنگ سفید و مقدار ۱ برای مشخص شدن و تشخیص ویژگی ها توسط شبکه کانولوشنی بصورت دقیقتر.

لایه های استفاده شده در این شبکه که در بالا بیان شد را در این قسمت بصورت خلاصه شرح میدهیم:

- لایه کانولوشنی: در این شبکه از این لایه برای استخراج ویژگی های تصویر با تعداد فیلتر های گوناگون استفاده شده است که تعداد فیلتر ها و ابعاد آن ها در ۴ بلوک کانولوشنی معرفی شده در جدول ۱ آورده شده است.
- لایه Max-pooling : بین دو لایه کانولوشن قرار می گیرد تا ابعاد تصویر کاهش یابد. علاوه بر این، عملیات ادغام پیچیدگی محاسباتی شبکه را به حداقل می رساند. لایه ادغام همچنین از فرآیند انتخاب ویژگی و کاهش اضافه برازش پشتیبانی می کند. در ادغام، ویژگی های استخراج شده در یک منطقه خاص مورد استفاده قرار می گیرد و در مناطق دیگر نیز قابل استفاده است. معمولاً از اندازه فیلتر ۲*۲ برای عملیات جمع آوری حداکثر استفاده می شود. ترکیب لایه‌های کانولوشنال و تلفیقی به طور مکرر استفاده می‌شود تا زمانی که در نهایت با برداری کار کند که بتواند آن را مدیریت کند.
- لایه Dropout و Batch Normalization : لایه Dropout برای جلوگیری از **Overfitting** یا بیش برازش قرار داده میشود تا از پیچیدگی مدل بکاهد و مدل داده هارا حفظ نکند و عمومیت مدل بالا رود.

این مدل با ساخت یک CNN معمولی با ۱ بلوک شروع می‌شود و سپس عمیق‌تر می‌شود و از ترکیب‌های فراپارامترهای مختلف برای عملکرد بهتر استفاده می‌کند. در تمام لایه های کانولوشنال شبکه DCNN، نمایش ورودی واقعی با استفاده از padding یکسان انجام میشود. بلوک ۱ استخراج ویژگی‌های کانولوشنال با اولین لایه کانولوشن آغاز میشود که دارای ۳۲ نقشه ویژگی است که هر کدام دارای اندازه فیلتر قابل آموزش (۳*۳) پیکسل و یک تابع فعال‌سازی (ReLU) است. این می تواند ویژگی ها را از تصویر خام

ورودی در اندازه ۴۰*۴۰ استخراج کند. لایه نرمال سازی دسته ای لایه دوم در بلوک اول است که درست بعد از لایه های کانولوشنال متصل شده است. برای کاهش ابعاد نقشه های ویژگی، از یک لایه ترکیبی به اندازه ۲*۲ به عنوان لایه سوم استفاده شد. برای کاهش overfitting، پس از ادغام لایه‌ها در هر بلوک، از یک لایه حذفی استفاده شده است. به طور مشابه، ۳ بلوک لایه های پیچیدگی دیگر برای تکمیل پیکربندی مدل انباشته شده اند. پس از تکمیل بلوک های کانولوشنال، بلوک لایه های طبقه بندی، شبکه را با لایه flatten آغاز می کند تا ماتریس ویژگی های دو بعدی را به یک بردار تبدیل کند. در نهایت، آن را به لایه FC 1 با ۱۰۲۴ نورون متصل میکند. پس از آن یک لایه dropout دسته ای با مقدار حذف ۰.۵ متصل شده است. در نهایت، یک لایه FC دوم در انتهای شبکه با فعالساز softmax اعمال میشود که خروجی را میدهد.

## ۳-۴. نمودار های خطا و صحت و معیار های درستی

مدل DCNN مطرح شده در بالا را به تعداد epoch = ۳۰ با Batch size = 64 بر روی حدود ۶۰۰۰۰ داده آموزشی بوسیله سه الگوریتم بهینه سازی Momentum, Adam, Adadelta آموزش دادیم.

داده های تست را با نسبت ۰.۳ به دو بخش validation و test تقسیم کرده و از داده های val در طول آموزش برای صحت سنجی مدل و توقف مدل برای جلوگیری از overfitting و همچنین تنظیم هایپر پارامتر ها استفاده شده و همچنین در آخر نیز بهترین مدل بدست آمده بر اساس بیشترین مقدار val accuracy، ذخیره و بر روی ۲۰۰۰۰ داده تست امتحان شد و مقادیر و متریک های مختلف برای آن بدست آمد که در هر الگوریتم بصورت زیر میباشد:

- ***بهینه سازی Momentum:***

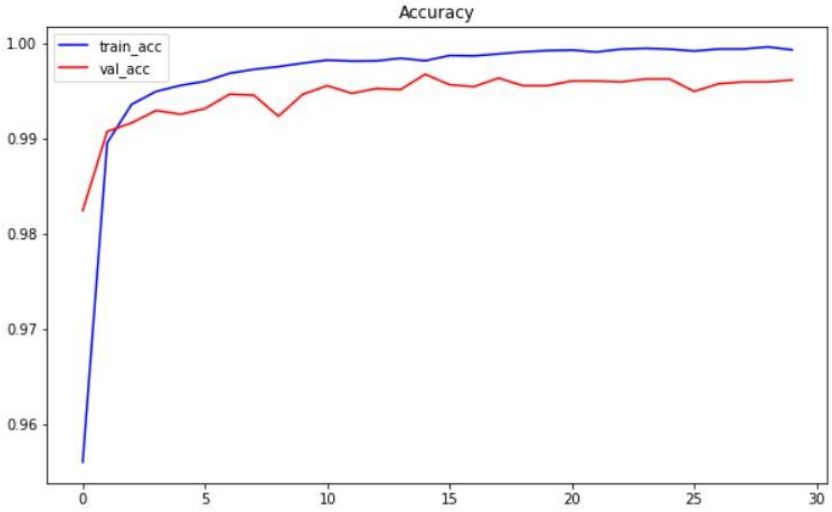
نتایج آخرین epoch:

```
Epoch 30/30
937/937 [==============================] - 29s 31ms/step - loss: 0.0022 - accuracy: 0.9993 - val_loss: 0.0181 - val_accuracy:
0.9961

Epoch 00030: val_accuracy did not improve from 0.99670
```

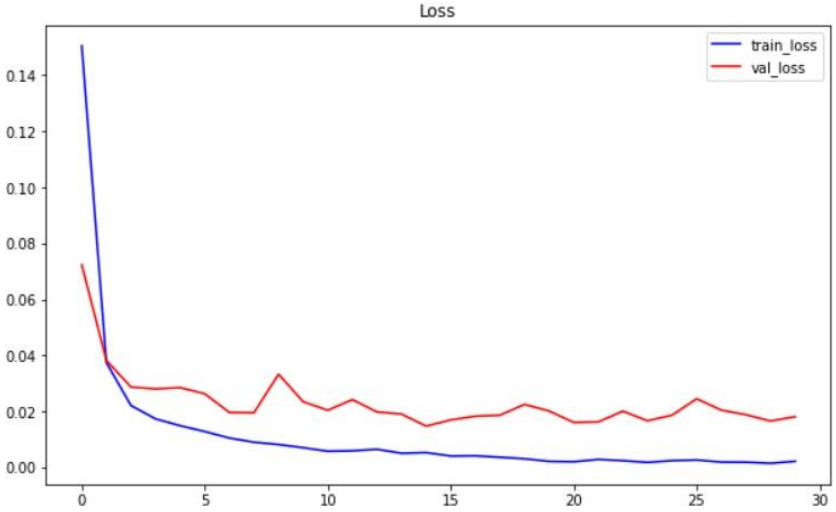
نتایج بر روی داده های val :

```
model evaluation :
313/313 [==============================] - 2s 7ms/step - loss: 0.0181 - accuracy: 0.9961
validation accuracy : 99.610
```

- نمودار loss :




- نمودار accuracy :




- مقادیر Recall, Percision, F1 Score و Confusion matrix :

```
Percision Score : 0.99625

Recall Score : 0.99625

F1 Score : 0.99625

Confusion Matrix :

[[1997    0    0    0    1    2    0    0    0    0]
 [   2 1991    0    0    1    0    0    0    1    5]
 [   1    2 1987    6    2    0    0    0    0    2]
 [   0    0    9 1982    8    1    0    0    0    0]
 [   0    0    2    6 1992    0    0    0    0    0]
 [   3    0    0    0    1 1996    0    0    0    0]
 [   0    0    0    0    2    4 1990    0    0    4]
 [   1    1    2    0    1    0    1 1994    0    0]
 [   0    0    0    0    0    0    0    0 1999    1]
 [   0    1    0    0    0    1    1    0    0 1997]]
```

- ***بهینه سازی Adam:***

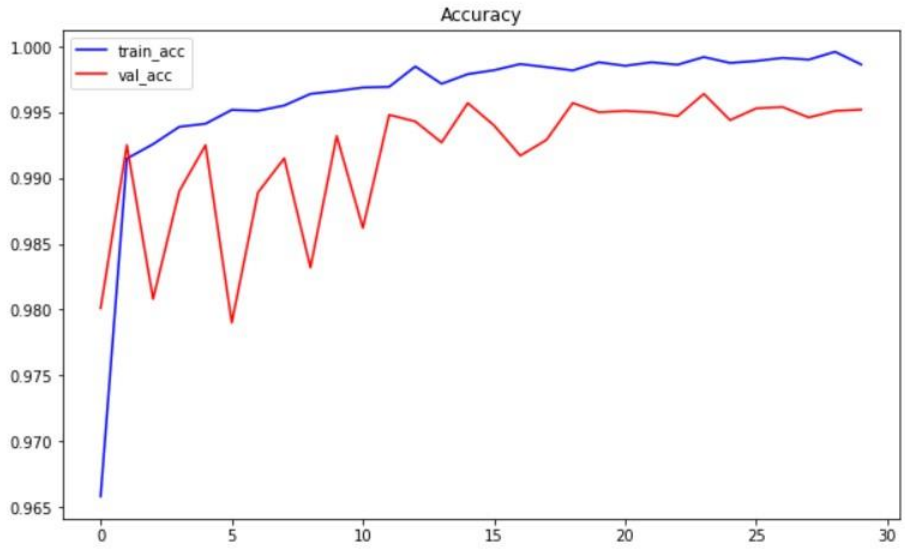
نتایج آخرین epoch:

```
Epoch 30/30
937/937 [==============================] - 32s 34ms/step - loss: 0.0057 - accuracy: 0.9986 - val_loss: 0.0284 - val_accuracy:
0.9952

Epoch 00030: val_accuracy did not improve from 0.99640
```

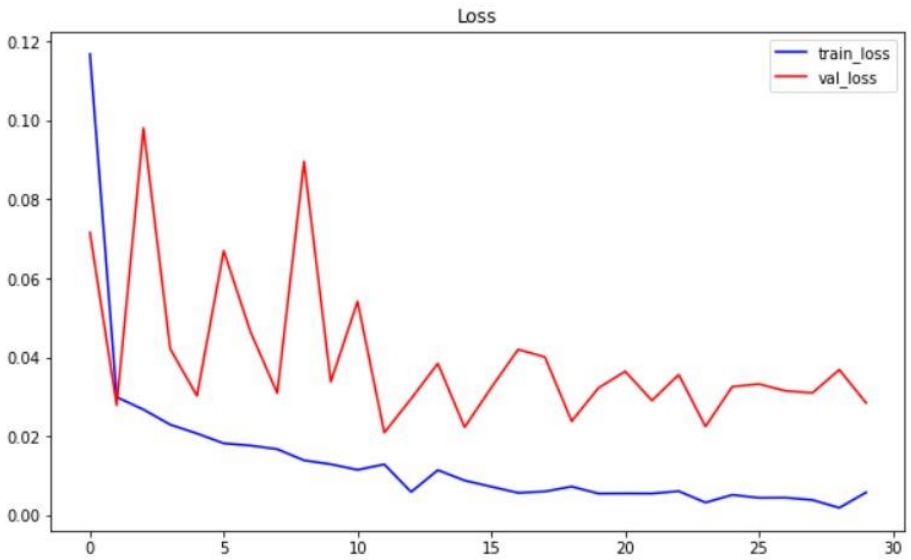
نتایج بر روی داده های val :

```
model evaluation :
313/313 [==============================] - 3s 8ms/step - loss: 0.0284 - accuracy: 0.9952
validation accuracy : 99.520
```

- نمودار loss :




- نمودار accuracy :



- مقادیر Recall, Percision, F1 Score و Confusion matrix :

```
Percision Score : 0.99635

Recall Score : 0.99635

F1 Score : 0.99635

Confusion Matrix :

[[1995    0    0    0    2    0    1    2    0    0]
 [   1 1996    0    0    1    0    0    0    1    1]
 [   0    2 1988    4    4    0    0    0    0    2]
 [   0    0   11 1986    2    1    0    0    0    0]
 [   1    0    2    2 1994    0    1    0    0    0]
 [   3    2    1    0    1 1993    0    0    0    0]
 [   0    0    1    0    0    5 1991    0    0    3]
 [   0    3    2    1    0    0    0 1994    0    0]
 [   0    0    0    0    0    0    0    0 2000    0]
 [   2    3    0    0    0    1    4    0    0 1990]]
```

- ***بهینه سازی Adadelta:***

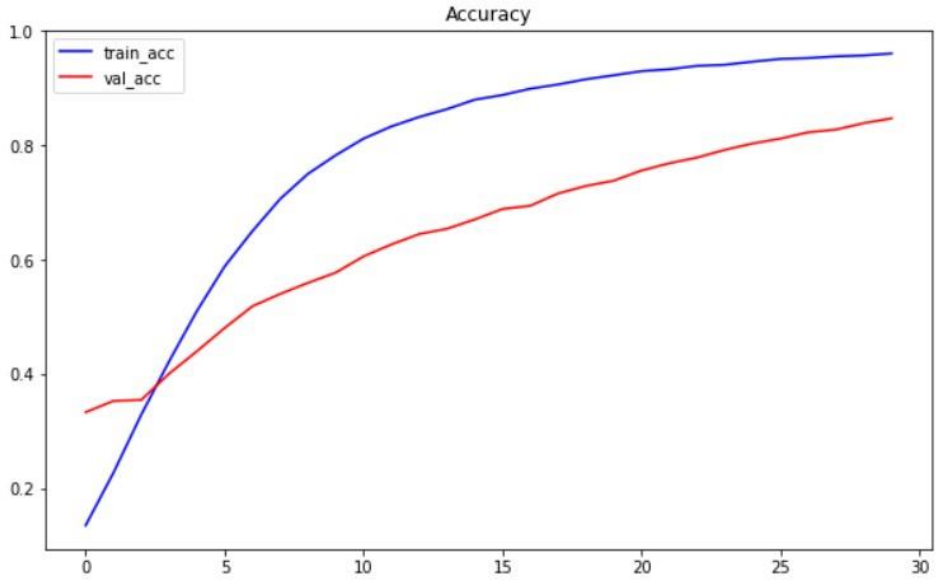
نتایج آخرین epoch:

```
Epoch 30/30
937/937 [==============================] - 31s 33ms/step - loss: 0.1266 - accuracy: 0.9602 - val_loss: 0.5110 - val_accuracy:
0.8467
```

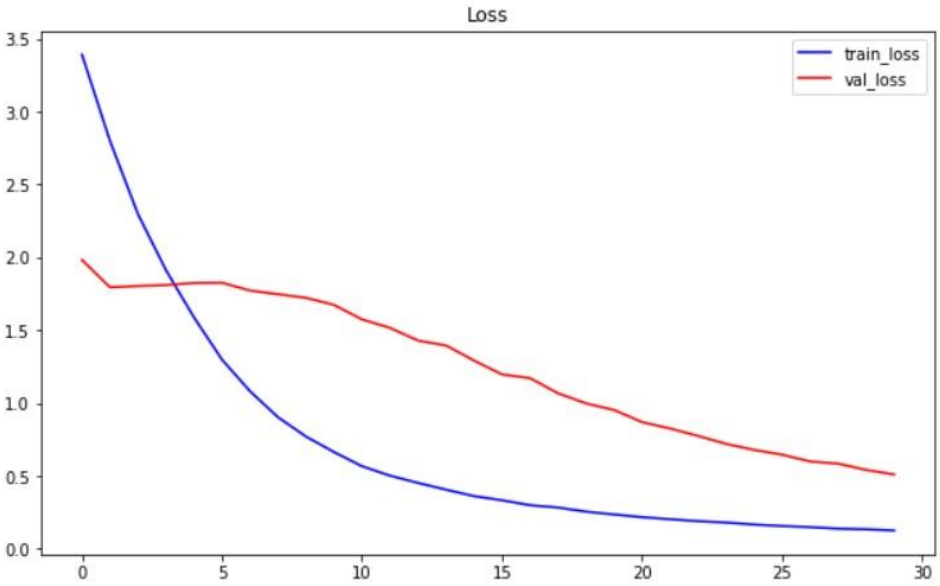
نتایج بر روی داده های val :

```
model evaluation :
313/313 [==============================] - 2s 7ms/step - loss: 0.5110 - accuracy: 0.8467
validation accuracy : 84.670
```

- نمودار loss :

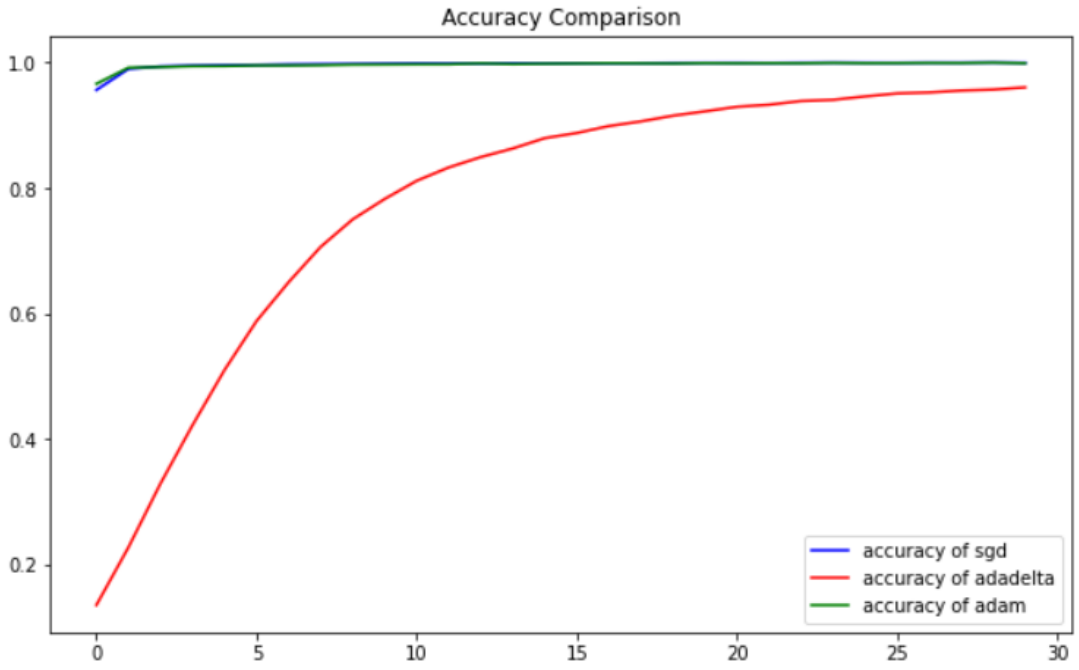


- نمودار accuracy :




- مقادیر Recall, Percision, F1 Score و Confusion matrix :

```
Percision Score : 0.84455

Recall Score : 0.84455

F1 Score : 0.84455

Confusion Matrix :

[[1993    4    0    0    0    0    0    2    1    0]
 [   6 1994    0    0    0    0    0    0    0    0]
 [   0  120 1876    0    0    0    0    4    0    0]
 [  20    4 1023  941    2    0    0    9    1    0]
 [  37    7  341   36 1569    0    8    0    0    2]
 [ 114   45    4    0   13 1105    4    1  714    0]
 [  34   31   83    0    1    1 1815    5    3   27]
 [  13   22   30    0    0    0    0 1935    0    0]
 [   2   29    0    0    0    0    0    0 1969    0]
 [  46  159   24    0    0    0   47    0   30 1694]]
```




با توجه به نمودار های بالا میتوان به نکات زیر توجه کرد:

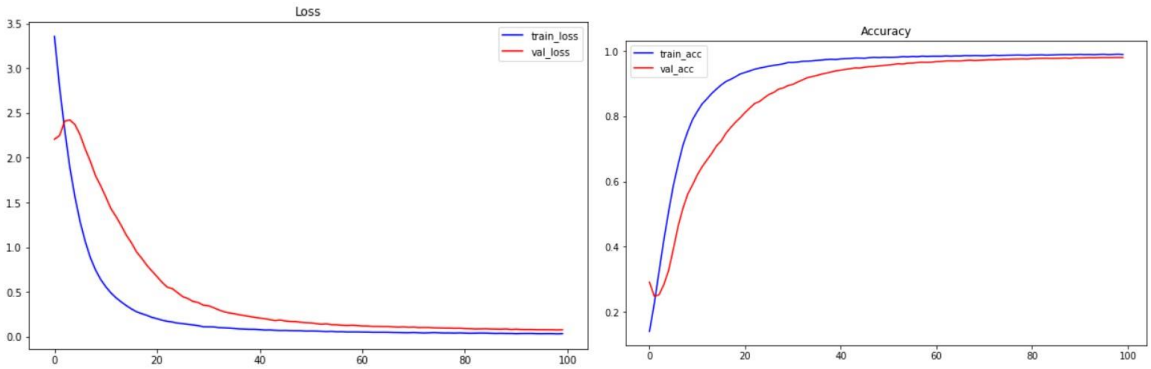
- الگوریتم Adadelta دارای نمودار نرم تری نسبت به بقیه میباشد، اما سرعت همگرایی آن نسبت به بقیه کمتر است و در epoch های بیشتری به accuracy بالا تر میتوان دست یافت.
- الگوریتم Adam دارای نوسانات زیادی در هنگام آموزش بر روی داده validation دارد ولی در epoch های مختلف طبق نمودار مربوطه از نوسانات آن کمتر میشود.
- بهترین الگوریتم بهینه سازی برای آموزش شبکه روش momentum میباشد که بر روی SGD پیاده شده است. این روش دارای همگرایی بالا و دقتی نزدیک به accuracy داده های آموزش دارد و از طرف دیگر نیز بین دو الگوریتم دیگر دقت نهایی آن در یک epoch مشخص بیشتر است.

برای epoch = ۱۰۰ مدل را آموزش دادیم:

**Adadelta:**

```
Epoch 100/100
937/937 [==============================] - 30s 32ms/step - loss: 0.0330 - accuracy: 0.9896 - val_loss: 0.0756 - val_accuracy:
0.9804

Epoch 00100: val_accuracy did not improve from 0.98060
```

```
model evaluation :
313/313 [==============================] - 2s 7ms/step - loss: 0.0756 - accuracy: 0.9804
validation accuracy : 98.040
```

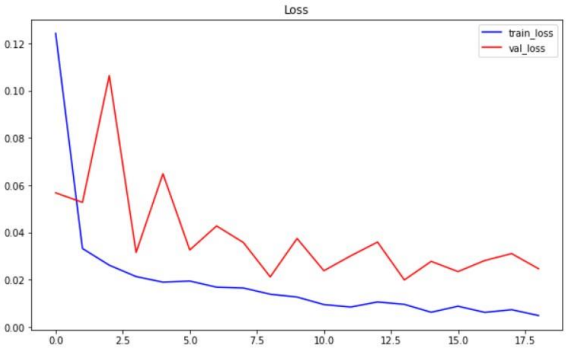


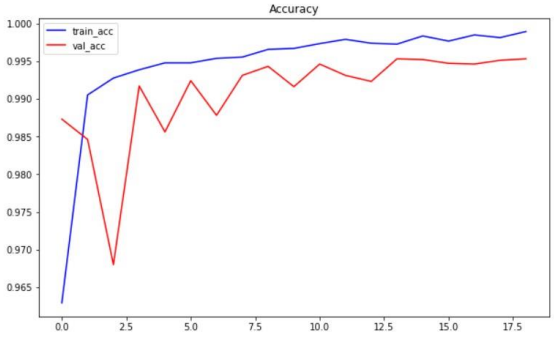

```
Percision Score : 0.97855

Recall Score : 0.97855

F1 Score : 0.97855

Confusion Matrix :

[[1999    0    0    0    0    0    0    1    0    0]
 [   0 2000    0    0    0    0    0    0    0    0]
 [   2   30 1963    0    1    0    1    1    0    2]
 [   7    1   99 1879   12    0    0    1    1    0]
 [   6    2   13   11 1962    0    4    0    0    2]
 [  66   11    2    0    3 1867    3    0   46    2]
 [   4    6    1    0    0    1 1973    0    1   14]
 [   4   11    5    0    1    0    0 1979    0    0]
 [   0   17    0    0    0    0    0    0 1982    1]
 [   5   20    0    0    0    1    7    0    0 1967]]
```

## Adam:

```
Epoch 19/50
937/937 [==============================] - 30s 32ms/step - loss: 0.0049 - accuracy: 0.9989 - val_loss: 0.0247 - val_accuracy:
0.9953

Epoch 00019: val_accuracy did not improve from 0.99530
Epoch 00019: early stopping
```

```
model evaluation :
313/313 [==============================] - 2s 7ms/step - loss: 0.0247 - accuracy: 0.9953
validation accuracy : 99.530
```



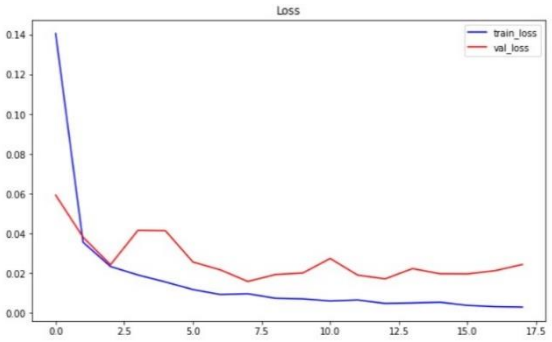



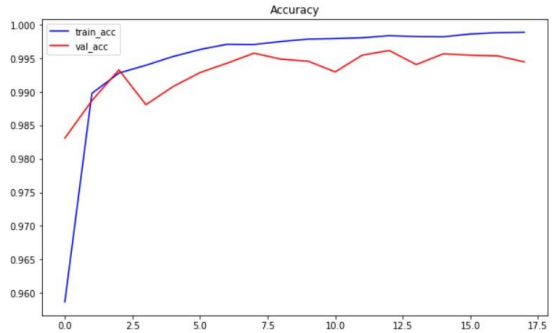

```
Percision Score : 0.99515

Recall Score : 0.99515

F1 Score : 0.99515

Confusion Matrix :

[[1990    0    0    0    2    6    1    1    0    0]
 [   1 1994    0    0    1    0    0    0    1    3]
 [   2    3 1973   15    4    0    0    1    0    2]
 [   1    0    6 1987    4    1    0    1    0    0]
 [   0    0    2    1 1991    4    2    0    0    0]
 [   2    0    0    0    1 1997    0    0    0    0]
 [   4    0    0    0    0    3 1986    3    0    4]
 [   1    0    2    0    0    3    0 1994    0    0]
 [   0    1    0    0    0    0    0    0 1999    0]
 [   1    2    0    0    0    2    2    1    0 1992]]
```

**SGD:**

```
Epoch 18/50
937/937 [==============================] - 29s 31ms/step - loss: 0.0030 - accuracy: 0.9989 - val_loss: 0.0244 - val_accuracy:
0.9945

Epoch 00018: val_accuracy did not improve from 0.99620
Epoch 00018: early stopping
```

```
model evaluation :
313/313 [==============================] - 2s 7ms/step - loss: 0.0244 - accuracy: 0.9945
validation accuracy : 99.450
```




```
Percision Score : 0.99595

Recall Score : 0.99595

F1 Score : 0.99595

Confusion Matrix :

[[1994    0    0    0    2    4    0    0    0    0]
 [   2 1994    2    0    0    0    0    0    0    2]
 [   0    1 1981   12    5    0    0    0    0    1]
 [   0    0    5 1988    7    0    0    0    0    0]
 [   0    0    2    3 1994    1    0    0    0    0]
 [   4    0    0    0    2 1994    0    0    0    0]
 [   0    2    1    0    1    4 1987    0    0    5]
 [   1    1    1    0    0    0    1 1996    0    0]
 [   0    1    0    0    0    0    0    0 1998    1]
 [   1    5    0    0    0    1    0    0    0 1993]]
```

## ۳-۵. معماری و پارامتر های بهترین شبکه

بهترین الگوریتم و معماری استفاده شده بوسیله momentum میباشد که دارای هاپیرپارمتر تکانه 0,9 و مقدار learning rate = 0.01 میباشد.