گزارش مینی‌پروژه اول – شبکه عصبی

میثم پرویزی
دانشکده مهندسی برق و کامپیوتر
پردیس دانشکده‌های فنی
دانشگاه تهران

سوال ۱: بررسی روش‌های بهینه‌سازی تابع زیان

بخش (a): **مشکلات روش Gradient Descent**

۱. همگرا شدن به یک مینیمم محلی می‌تواند بسیار کند باشد.
۲. به مقادیر اولیه وزن‌ها وابسته است.
۳. ممکن است در یک مینیمم محلی گیر کند و نیاز به شروع مجدد داشته باشد.
۴. در انتخاب Learning Rate باید دقت کافی به خرج داده شود.

بخش (b): **معرفی روش‌های جایگزین Gradient Descent**

در روش Gardient Descent از رابطه‌ی زیر برای بهینه‌سازی وزن‌ها و بایاس‌ها استفاده می‌شود که موجب مشکلات فوق می‌شود لذا روش‌های دیگری برای بهینه‌سازی معرفی شدند تا این مشکلات را کاهش دهند.

$$\Delta \mathrm{L} = \left(\frac{\partial \mathrm{L}}{\partial \mathrm{w}}, \frac{\partial \mathrm{L}}{\partial \mathrm{b}}\right)$$

$$\mathrm{w_k} \leftarrow \mathrm{w_k} - \alpha \frac{\partial \mathrm{L}}{\partial \mathrm{w_k}}$$

$$\mathrm{b_k} \leftarrow \mathrm{b_k} - \alpha \frac{\partial \mathrm{L}}{\partial \mathrm{b_k}}$$

۱. **روش SGD+Momentum**

در این روش برای آپدیت کردن وزن‌ها و بایاس‌ها از یک رابطه استفاده می‌شود که از قوانین فیزیک الهام گرفته شده است. فرض کنید یک توپ در یک زمین با پستی و بلندی‌های فراوان در حال حرکت به سمت عمیق‌ترین دره است. زمانی که شیپ یک تپه بسیار زیاد باشد، توپ تکانه‌ی بیش‌تری به خود می‌گیرد و می‌تواند از تپه‌های کوچک‌تر به راحتی عبور کند. هر چه شیب افزایش یابد، تکانه و سرعت توپ نیز افزایش خواهد یافت تا در نهایت در عمیق‌ترین نقطه‌ی دره متوقف شود.

این روش الگوریتم Gradient Descent را به این صورت اصلاح می‌کند که دو پارامتر جدی‌د به رابطه‌ی بهینه‌سازی اضافه می‌کند. یکی **سرعت ν** که می‌خواهیم آن را بهینه کنیم و دیگری اصطکاک μ که تلاش می‌کند تا سرعت را کنترل کند تا از روی دره جهش نکنیم و در عین حال سرعت کاهش بیشتری داشته باشیم.

$$\nu = \mu\nu - \alpha\Delta L$$
$$w = w + \nu$$

۲. روش Adam

در روش‌های بهینه سازی که تا کنون بررسی کردیم همواره نرخ یادگیری α مقدار ثابتی داشت. اما در روش‌های تطبیقی (Adaptive) مقدار این پارامتر نیز می‌تواند در راستای بهینه‌سازی تغییر کند. روش Adam از مزایای روش‌های تطبیقی در کنار مزایای روش Momentum استفاده می‌کند تا سرعت و دقت همگرایی به نقطه‌ی بهینه را افزایش دهد. در این روش دو پارامتر β_1 و β_2 با مقادیر پیشنهادی 0.9 و 0.999 معرفی شده‌اند. همچنین در این روش مفهوم تکانه‌ی n ام یا n-th Moment به صورت امید ریاضی یک متغیر تصادفی به توان n تعریف می‌شود.

$$m_n = E[X^n]$$

که رابطه‌ی آپدیت وزن‌ها نیز به صورت زیر می‌باشد:

$$m_t = \beta 1 m_t - 1 + (1 - \beta 1)\Delta L$$
$$\nu_t = \beta 2 \nu_t - 1 + (1 - \beta 2)\Delta L^2$$

۳. روش AdaDelta

این روش که بر اساس روش AdaGrad ایجاد شده است تلاش می‌کند تا جلوی کاهش شدید و یکنواخت نرخ یادگیری را بگیرد. به جای انباشته کردن تمام مجذور گرادیان‌های قبلی، در این روش مجموع مجذورها گرادیان‌های قبلی به یک مقدار مشخص محدود می‌شود.

سوال ۲: معرفی توابع زیان مختلف

۱. Hinge loss

تابع اتلاف Hinge برای آموزش طبقه‌بندی کاربرد دارد. این تابع بیشتر برای حاشیه بیشینه طبقه‌بندی در ماشین‌های بردار پشتیبانی (SVMs) کاربرد دارد و برای خروجی $t = \pm 1$ و امتیاز طبقه‌بندی y از رابطه زیر محاسبه می‌شود:

$$\boldsymbol{L(y) = \max(0, 1 - t.y)}$$

۲. Softmax cross entropy

این تابع نیز برای طبقه بندی کاربرد دارد و ترکیبی از دو تابع softmax و cross entropy است به این صورت که خروجی softmax را به cross entropy می‌دهد. تابع softmax از رابطه زیر محاسبه می‌شود:

$$\boldsymbol{S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}}$$

و تابع cross entropy از رابطه زیر به دست می‌آید:

$$\boldsymbol{H(y, p) = -\sum_c y_c . log(p_c)}$$

۳. **Mean squared error**

یکی از معروف‌ترین و معمول‌ترین توابع زیان در تحلیل رگرسیونی، میانگین مربعات خطا (Means Square Error) است که به اختصار MSE نامیده می‌شود. این تابع زیان، میانگین مربعات فاصله بین مقدار پیش‌بینی و واقعی را محاسبه می‌کند. شیوه و نحوه محاسبه آن در زیر دیده می‌شود:

$$MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{n}$$

۴. **Log loss**

همان cross entropy است که توضیح داده شد و برای طبقه‌بندی مناسب است.

سوال ۳: آشنایی با Overfitting

در دنیای الگوریتم‌ها Overfit شدن به معنای این است که الگوریتم فقط داده‌هایی را که در مجموعه آموزشی(Train Set)یاد گرفته است را می‌تواند به درستی پیش‌بینی کند ولی اگر داده‌ای کمی از مجموعه‌ی آموزشی فاصله داشته باشد، الگوریتمی که Overfit شده باشد، نمی‌تواند به درستی پاسخی برای این داده‌های جدید پیدا کند و آن‌ها را با اشتباهِ زیادی طبقه‌بندی می کند.

روش‌های مختلفی برای مقابله با Overfitting وجود دارد که چند نمونه از آن‌ها عبارتند از:

Dropout: در این روش در حین فرآیند آموزش (Training) تعدادی از نورون‌ها را به طور تصادفی نادیده می‌گیریم. به این معنا که اثر آن‌ها بر نورون‌های بعدی موقتاً حذف می‌شود و در مسیر برگشت نیز وزن‌های نورون مورد نظر آپدیت نمی‌شود. به این ترتیب حساسیت کل شبکه نسبت به یک نورون خاص کاهش پیدا می‌کند و احتمال Overfitting را کاهش می‌دهد.

Norm Penalty: در روش‌های Norm Penalty هنگام آپدیت کردن پارامترهای شبکه، یک عبارت دیگر با عنوان Norm Penalty با آن جمع می‌کنیم تا از Overfitting جلوگیری کند.

Early Stopping: در این روش برای جلوگیری از Overfitting باید فرآیند آموزش را به محض آن که شبکه به دقت بسیار پایینی روی داده‌های تست برسد متوقف کنیم.

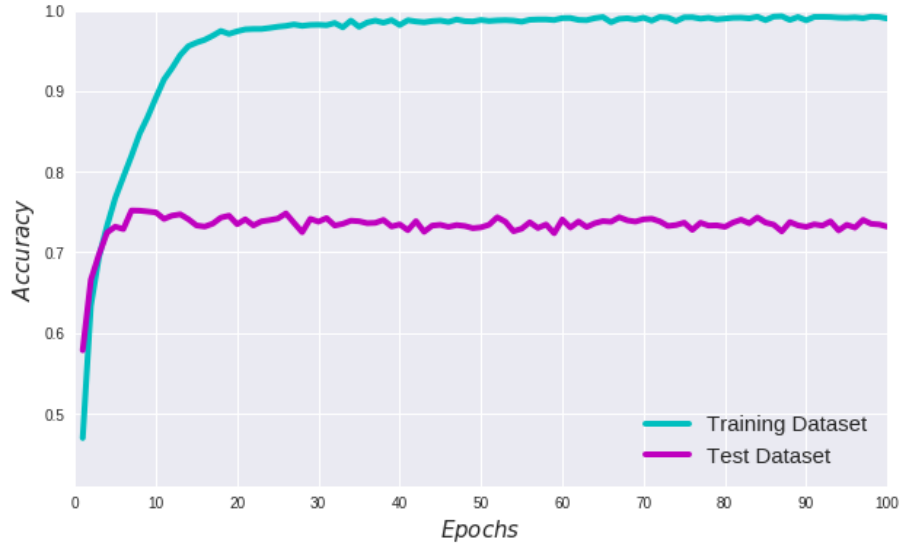
سوال ۴: پیاده‌سازی شبکه برای یادگیری داده‌های CIFAR10

بخش A: **مشخصات شبکه عصبی**

- اندازه پنجره کانولوشن: (۳،۳)
- اندازه stride: (۱،۱)
- تعداد فیلترها در هر لایه: ۱ عدد
- اندازه Max Pooling: (۲،۲)
- توابع فعال‌ساز: ReLU در طبقات میانی و SoftMax در خروجی
- تعداد و اندازه لایه‌های Fully Connected: ۱ عدد با ۲۵٦ نورون
- تابع Loss: Cross Entropy

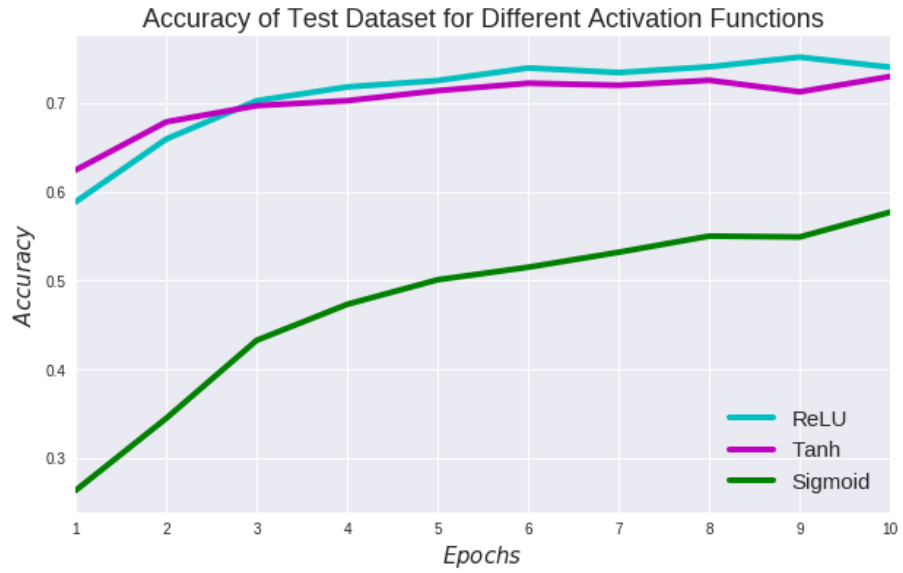
- روش بهینه‌سازی: Adam
- اندازه Mini-Batch: ۱۰۰

بخش B: **نمودار Accuracy بر حسب Epoch**



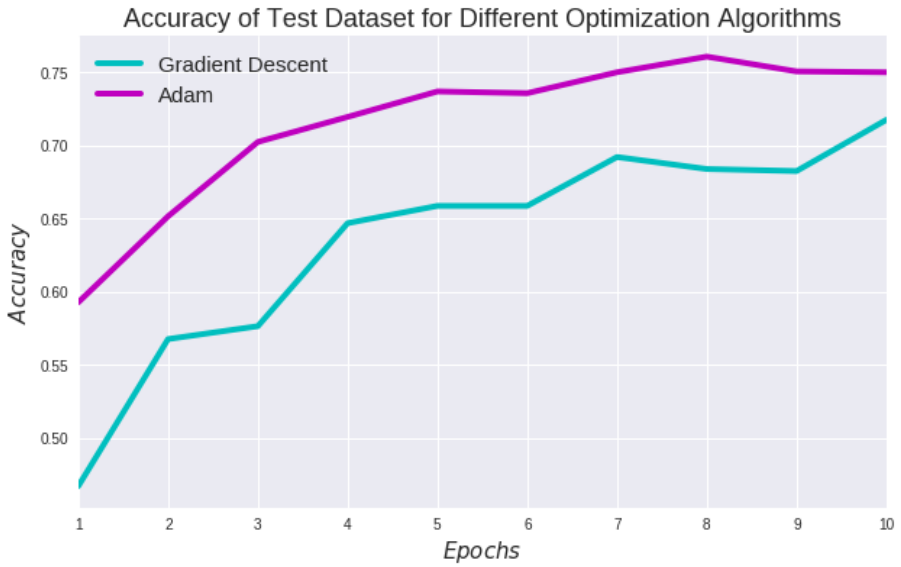
بهترین دقت برای داده‌های Train برابر با ۹۹ درصد است که بعد از حدوداً ۴۰ ایپاک و برای داده‌های تست ۷۳ درصد است و بعد از حدوداً ۱۰ ایپاک به دست می‌آید.

بخش C: **بررسی عملکرد توابع فعال‌ساز مختلف**



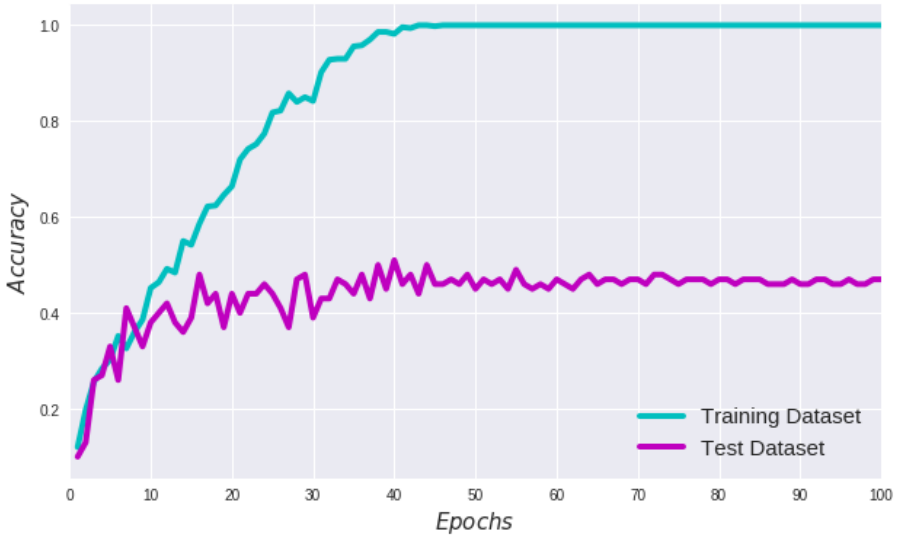
تابع ReLU عملکرد بهتری دارد چون برخلاف تابع Sigmoid یا Tanh گرادیان آن اشباع نمی‌شود و می‌توان سریع‌تر منجر به همگرایی شود.

بخش D: **بررسی روش‌های بهینه‌سازی مختلف**



همان‌طور که در بخش‌های قبلی گفته شد روش Adam به علت استفاده از مفهوم Momentum می‌تواند سریعتر و به نقطه‌ی درست‌تری همگرا شود و در نتیجه به دقت بالاتری دست می‌یابد.

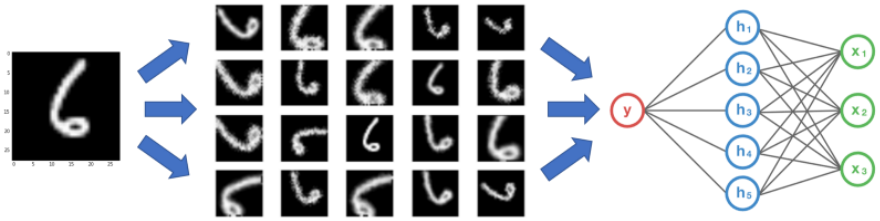
بخش E: **کاهش حجم داده‌ها**



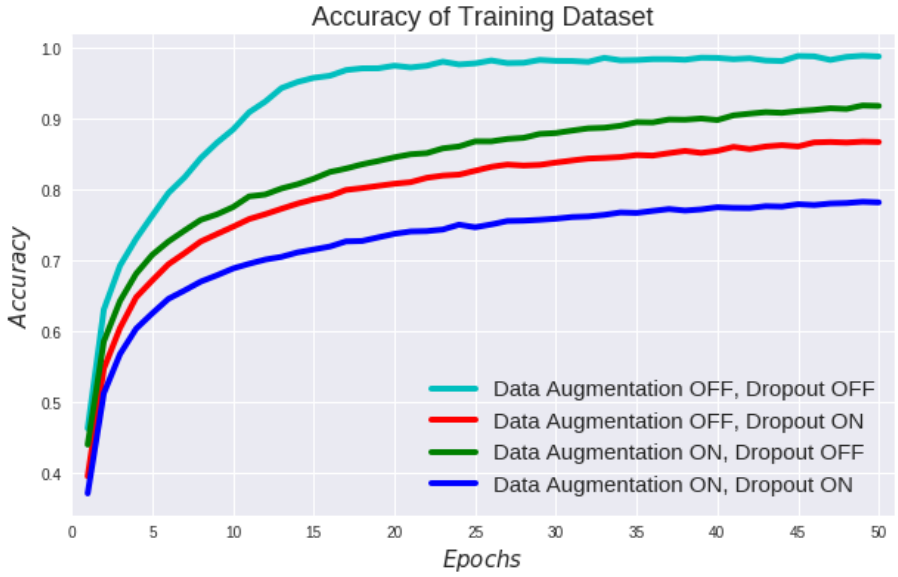
حجم داده‌های Train را به ۵۰۰ عدد و داده‌های تست را به ۱۰۰ عدد (مجموعاً ۶۰۰ عدد) کاهش دادیم و فرایند آموزش را انجام دادیم. مشاهده می‌شود که در مقایسه با بخش B تفاوتی چندانی در نتایج Train ایجاد نشده است اما در نتایج Test مشاهده می‌شود که دقت به حدود ۴۷ درصد کاهش یافته است. این نشان می‌دهد که برای بالا بردن دقت شبکه باید تعداد داده‌های Train زیاد باشد.

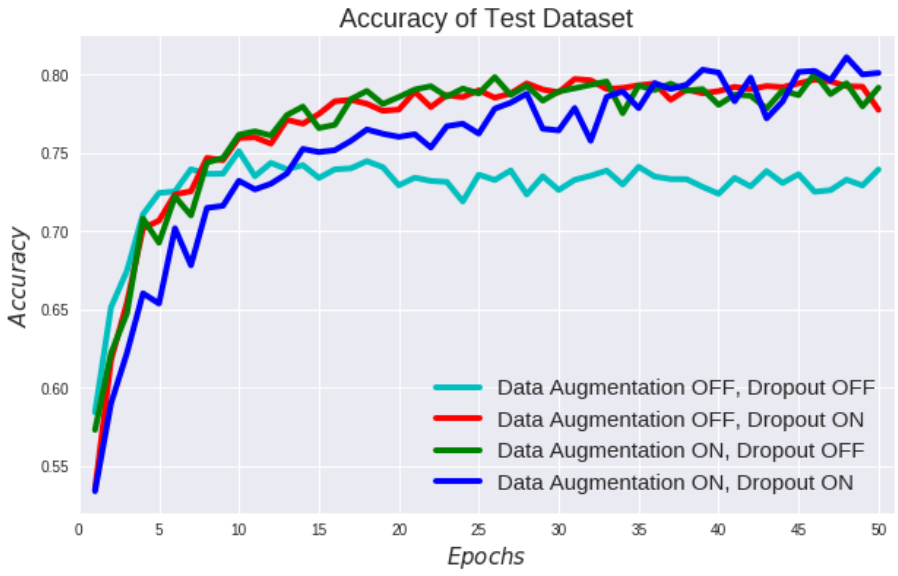
بخش F: **بررسی تاثیر Dropout و Data Augmentation**

(a) تکنیک Data Augmentation به این معناست که ما با داشتن همان داده‌های Train اولیه و صرفا با اعمال چندین تبدیل رو داده‌های ورودی مثل چرخاندن تصاویر، بزرگنمایی، جابجایی و... تعداد داده‌های Train را افزایش دهیم تا عمومیت شبکه افزایش پیدا کند.



(b) پس از افزودن Dropout و Data Augmentation به شبکه نمودارهای Accuracy بر حسب Epoch را برای داده‌های Train و Test رسم کردیم.





همانطور که از روی نمودارها پیداست، به کار بردن هر کدام از این دو تکنیک روی دقت داده‌های Train اثر معکوس دارد اما در مورد داده‌های Test این طور نیست بلکه باعث افزایش دقت خروجی شبکه عصبی می‌شود. این موضوع در مورد تمام روش‌های Regularization در مقابله با Overfitting صادق است که همگی موجب افزایش دقت داده‌های Test می‌شوند. علت این موضوع آن است در هنگام تنظیم پارامترهای شبکه در روش‌های Regularization سعی می‌شود تا تاثیر داده‌های Train کاهش پیدا کند تا عمومیت شبکه افزایش یابد لذا انتظار می‌رود تا دقت خروجی شبکه در مورد داده‌های Train نیز کاهش پیدا کند

مراجع:

https://deepnotes.io/sgd-momentum-adaptive

http://cs231n.github.io/neural-networks-3/

https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/