گزارش تمرین 3

سحر برکاتی 96222021

میدانیم که سیستم بینایی ما وظیفه ، تشخیص اشیا را در مسیر ventral را دارد . از لایه V1 شروع میکنیم که وظیفه شناسایی خطوط و زوایا را دارد و هرچه به لایه های بالاتر میرویم میبینیم که دو تا اتفاق رخ میدهد :

1- Feature complexity افزایش می یابد.
2- Receptive field بزرگتر میشود .

شبکه های عصبی عمیق کانولوشنی ، که به آن DCNN هم گفته میشود ، منجر به ساخت مدل با عملکرد بالا مثل انسان شده است .

درDCNN برای این که هر بخش ، بخش دیگر را فعال کند نیاز است که به یک دیگر floating point ارسال کنند که این همانند pulse فرستادن در نورون های بیولوژیکی است.

این شبکه ها با الگوریتم back – propagation کار میکنند ولی بر خلاف نورون های بیولوژیکی همگرایی کند تری دارند . به دلیل این که DCNN ها میلیون ها پارامتر آزاد دارند، برای جلوگیری از این که over – fitting رخ دهد ، از داده های برچسب دار استفاده میکنیم . با این حال بیشتر مواقع داده های برچسب دار در اختیار ما نیست ، و میتوانیم با کمک قاعده STDP که یک قاعده یادگیری unsupervised است ، قادر به انجام این کار باشند . ما DCNN را بر اساس spike – rate کدگذاری میکنیم . ولی اشکالی که دارد این است که نیاز به سنسور های زیادی در پردازش های طولانی دارد. و بعلاوه این که استفاده از الگوریتم back – propagation و داشتن هر دو خروجی excitatory و inhibitory از نظر بیولوژیکی قابل قبول نیست. از طرفی SNN ها شبکه هایی بر اساس spike هستند و spiking pattern ها را یاد می گیرند.

شبکه های SDNN ، کدگذاری ای بر اساس spike time neural دارند . این شبکه ها لایه ای از temporal coding و بعد از آن لایه convolution ای و سپس لایه pooling را در بردارد . اتفاقی که رخ میدهد این است که آخرین لایه convolution ای ما هم یاد میگیرد و هم اشیا را به خوبی تشخیص میدهد .

یکی از معماری ها برای SDNN این است که سه لایه convolution ای به همراه سه لایه pooling داشته باشیم . اگر بخش بخش بخواهیم این معماری را بررسی کنیم داریم :

- در لایه اول ما از فیلتر DoG استفاده کردیم ، که این فیلتر تفاوت ها را در تصویر به ما نشان میدهد . و شدت این تفاوت ها در تاخیر spike های خروجی کد میشود .
- در لایه های convolution ، با استفاده از spike هایی که از لایه قبل که ویژگی های ساده ای را تشخیص داده اند می آید ، میتواند ویژگی های پیچیده تری را تشخیص دهند . ای لایه ها به محض این که ویژگی مورد نظر خود را مشاهده کنند ، یک spike تولید میکنند.
- در لایه های pooling ، هدف این است که با استفاده از فشرده سازی اطلاعات، حساسیت به ویژگی ها کاهش پیدا کند.

حالا میخواهیم لایه به لایه این معماری را بررسی کنیم :

لایه اول : نقش این لایه کد کردن سیگنال های ورودی برحسب زمان است . در این لایه ما یک فیلتر DoG روی Receptive field داریم . DoG ، ویژگی center – surround که در سلول های ganglion شبکیه وجود دارند

را شبیه سازی میکند . این سلول ها بعد از دیدن تصویر ، تضاد ها را تشخیص میدهد و یک spike تولید میکند . با rate – coding هرجا که پیکسل ما خروجی قوی تری داشت تعداد pulse بیشتر خواهد بود . DoG دو حالت on – center و off – center را انجام میدهد که به ترتیب به تضاد های مثبت و منفی حساس هستند . در واقع در اینجا حداکثر یکی از دو سلول چه مثبت و چه منفی میتواند در هر جایی fire کنند . در واقع اگر خروجی فیلتر DoG در مکان خاص r باشد بنابراین تایم firing متناظر با آن برابر با $1/r$ خواهد بود .

لایه های convolutional : هر لایه convolutional شامل چندین neuronal map است. برای هر نورون چند ویژگی تعریف میشود که ، با وزن های ورودی سیناپسی تعیین میشوند . نورون های هر بخش کل تصویر را میبینند و ویژگی مورد نظر را در مکان های مختلف بررسی میکنند. هر نورون موجود در این لایه spike های ورودی از لایه قبل را دریافت میکنند ، بر این حساب یک ویژگی بصری در لایه convolutional ترکیبی از چند ویژگی ساده در لایه های قبلی است . باید به این نکته توجه داشت که پنجره های ورودی دو نورون مجاور باهم over lapped دارند و به همین دلیل است که شبکه میتواند ظاهر ویژگی های بصری را در هر مکان تشخیص دهد . نورون های در همهء لایه های convolutional نورون های $nonleaky\ integrate\ and\ fire$ هستند که بدین صورت کار میکنند که spike های ورودی نورون های pre – synaptic را جمع میکنند و وقتی پتانسیل داخلی آن ها به threshold رسید ، spike میزنند و هر spike نورون های pre – synaptic با وزن های سیناپسی ، پتانسیل نورون را افزایش میدهند . و در هر بخش پتانسیل نورون i به این صورت update میشوند :

$$V_i = V_i(t-1) + \sum_j W_{ij} S_j(t-1)$$

V_i پتانسیل درونی نورون i ام برای لایه convolutional در لحظه t است. W_{ij} وزن بین نورون i و j است ، S_j ، قطار پالسی نورون j است . اگر V_i مقداری بیشتر از threshold یعنی V_{thr} بگیرد، پس V_i یک spike میزند و V_i مجددا به شکل زیرتنظیم میشود:

$$V_i(t) = 0 \quad and \quad S_j(t) = 1 \qquad if\ V_i(t) \geq V_{thr}$$

با وجود مکانیزم مهاری ای که در لایه convolutional وجود دارد ، هنگامی که یک نورون در مکان خاصی spike میزند ، نورون های دیگر را در آن مکان که متعلق به صفحه های دیگر را مهار میکنند و اجازه نمیدهد که تصویر بعدی را نشان داده شود . و باید به این نکته توجه داشت که نورون ها بیش از یکبار مجاز به firing نیستند .

لایه های $pooling$: این لایه ها با عملیات max pooling غیرخطی روی مجموعه نورون هایی که در یک همسایگی هستند سعی میکند که شبکه را invariance نگه دارد . با استفاده از کد گذاری rank – order در شبکه عملیات ماکسیمم گیری در لایه های $pooling$ به سادگی شامل propagation اولین spike زده شده میشود. نورون های این لایه نورون های integrate and fire هستند که وزن های سیناپس های ورودی و threshold روی 1 تنظیم شده است. پس اولین spike آنها را فعالکرده و منجر به spike خروجی میشود . با استفاده از کد گذاری rank – order هر نورون لایه $pooling$ فقط یکبار مجاز به firing هستند . و باید به این نکته توجه کرد که در این لایه learning صورت نمیگیرد.کار دیگر این لایه فشرده سازی اطلاعات بصری است . به دلیل ماکسیمم گیری در این لایه ، نورون های مجاور با ورودی هایی که باهم over – lapped دارند، اطلاعات اضافی ای را دارند . به همین دلیل ، یک over – lapped بسیار کوچک بین پنجره های ورودی دو نورون لایه $pooling$ وجود دارد که تا هم اطلاعات اضافی را از بین ببرد و همچنین عمل فشرده سازی رخ دهد .

قاعده یادگیری براساس STDP : تا اینجا متوجه شدیم که یادگیری فقط در لایه های convolutional رخ میدهد چرا که با یادگیری ویژگی های ساده استخراج شده میتواند ویژگی های بصری را تشخیص دهد . وقتی که یک تصویر جدید نمایش داده شود ، نورون های لایه convolutional و نورون هایی که زودتر fire کرده اند و باعث تحریک STDP

شده اند ، pattern های وروردی یاد میگیرند و باهم دیگر رقابت میکنند. یکی از ورژن های ساده STDP به فرم زیر است :

$$\Delta w_{ij} = \begin{cases} a^{+} w_{ij}(1 - w_{ij}), & if\ t_j - t_i \le 0 \\ a^{-} w_{ij}(1 - w_{ij}), & if\ t_j - t_i \ge 0 \end{cases}$$

که در اینجا i و j دو نورون $post - pre\ synaptic$ هستند. t_i و t_j زمان های spiking دو نورون i و j است . Δw_{ij} متغییری برای تغییرات وزن های سیناپسی است . دو پارامتر a^{+} و a^{-} ، نمایش دهنده learning rate هستند .اگر spiking time نورون $pre - synaptic$ به نورون $post - synaptic$ نزدیک باشد در نیجه آن تاخیر زمانی بسیار کم است .

ترم $(1 - w_{ij})$ وزن ها را بین [0,1] تنظیم میکند و به همین دلیل ، تمام سیناپس ها در حالت تحریکی حفظ میشوند .

- اگر برای a^{+} و a^{-} مقادیر بزرگ در نظر گرفته شود ، حافظه یادگیری کاهش میابد. یعنی فقط تصویر آخر را یاد میگیرند.
- اگر a^{+} و a^{-} مقادیر کوچک در نظر گرفته شوند ، روند یادگیری کند میشود . چون در ابتدا وزن های سیناپس ها رندوم انتخاب میشوند ، نورون ها الگو خاصی ندارند و به صورت تصادفی به الگو ها پاسخ میدهند ، و احتمال depression سیناپس بیشتر از تقویت شدن آن است.
- اگر $a^{-} > a^{+}$ ، وزن های سیناپسی کاهش پیدا میکنند تا حدی که دیگر به threshold نخواهند رسید .
- اگر $a^{+} > a^{-}$ ، نورون ها تمایل دارند بیشتر از یک الگو یادبگیرند و به آنها پاسخ دهند .

پس در نتیجه حالت آخر بهتر است .

در طول یادگیری ، نورون ها با شناسایی ویژگی ها در مکان های مختلف spike های ورودی را باهم ادغام میکند و برای STDP با یکدیگر رقابت میکنند. اولین نورونی که به threshold میرسد و fire میکند ، میتواند STDP را فعال کند و در نتیجه وزن های سیناپسی را update کند . وقتی یک نورون در یک صفحه درحال انجام STDP است، دیگر STDP نمیتواند برای نورون های دیگر در صفحات دیگر رخ دهد. در نتیجه ، این رقابت برای تشویق نورون های صفحات دیگر برای یادگیری ویژگی های مختلف بسیار مهم است. به دلیل گسسته بودن متغییر زمان در این مدل ، این احتمال وجود دارد که نورون های رقیب همزمان fire کنند. پس بهتر است که نورونی را انتخاب کنیم که بالاترین پتانسیل را دارد و نشان دهنده شباهت بیشتر بین ویژگی های آموخته شده و الگو ورودی آن است .
وزن های سیناپسی ، با مقدار های تصادفی آغاز میشود و از توزیع نرمال با $\mu = 0.8$ و $\sigma = 0.05$ پیروی میکند. و نباید مقدار μ *کوچک باشد* چون در نتیجه آن ممکن است نورون به threshold نرسد که پس learning رخ نخواهد داد. و اگر σ را بزرگ انتخاب کنیم، وزن های سیناپسی اولیه ممکن است بزرگتر باشند و سهم بیشتری در فعایت نورون ها دارد،و با توجه به قانون STDP همگرایی بیشتر به سمت 1 است .
ما همگرایی لایه I ام convolutional به این شکل است :

$$C_1 = \sum_{f} \sum_{i} w_{f,i}(1 - w_{f,i})/n_w$$

که $w_{f,i}$ یعنی وزن سیناپسی iام برای ویژگی fام و n_w مجموع تمام وزن های سیناپسی است . اگر وزن های سیناپسی یه سمت 0 یا 1 همگرا شوند ، C_1 به سمت 0 میل میکند. هرگاه C_1 به اندازه کافی نزدیک به 0 بود ، learning را متوقف میکنیم.

لایه $classification$: در این لایه نورون های لایه آخر یک $global\ \max pooling$ را روی صفحه مربوط به خود در آخرین لایه convolutional انجام میدهند . برای هر ویژگی ، یک مقدار خروجی وجود دارد که نشان دهنده وجود آن ویژگی در تصویر ورودی است . خروجی لایه $global\ \max pooling$، روی تصاویر دسته $train$ برای از طبقه بندی خطی SVM استفاده میشود. در مرحله آزمایش، تصویر نمایش داده شده ، پردازش میشود و خروجی $global\ \max pooling$ به $classifire$ برای تعیین دسته فرستاده میشود . برای محاسبه خروجی $global\ \max pooling$ ، ابتدا threshold لایه آخر convolutional را بینهایت در نظر میگیریم و سپس پتانسیل های نهایی آنها را اندازه گیری میکنیم. این پتانسیل را میتوان تعداد spike های اولیه مشترک ، بین ورودی جریان و نمونه های اولیه ذخیره شده در آخرین لایه convolutional در نظر گرفت.

حالا نتایج کاربرد این شبکه ها را داریم :

1- $face/motorbike\ dataset$:

ما دوتا دیتاست داریم ، یکی چهره و دیگری موتورسیکلت ، مجموعه آموزشی ما شامل 200تا دیتاست است. تمام تصاویر به مقیاس خاکستری تبدیل شده اند و ارتفاع آنها 160 پیکسل تغییر یافته است . ما از $SVM\ classifier$ استفاده میکنیم. شبکه ما دارای 3 لایه convolutional است .
لایه اول ، یک فیلتر DoG به اندازه 7×7 دارد، که ویژگی های بصری توسط این فیلتر محاسبه میشود. لایه های اول ، دوم ، سوم convolutional به ترتیب 4 و 20 و 10 صفحه با پنجره های convolutional ، 5×5 ، $16 \times 16 \times 4$ و $20 \times 5 \times 5$ است. و $threshold$ به ترتیب برابر با 10 ، 60 و 2 است.
سایز اولین و دومین لایه $pooling$ به ترتیب برابر با 7×7 و 2×2 و با اندازه $stride$ ، 6 و 2 . لایه سوم $pooling$ ، $global\ \max pooling$ را اجرا میکند. $a^+ = 0.004$ و $a^- = 0.003$.

در لایه اول convolutional ،برای ویژگی های بصری آموخته شده توسط یک نورون از تکنیک $backward\ reconstruction$ استفاده میکنیم که یعنی ویژگی لایه فعلی از ترکیب های وزنی ویژگی های لایه قبلی تشکیل شده. هر 4 صفحه این لایه ،به یکی از 4 زاویه $\frac{\pi}{4}$ ، $\frac{\pi}{2}$ ، $\frac{3\pi}{4}$ و π همگرا میشود . میبینیم که DoG و STDP لبه ها را تشخیص دادند و با شناسایی این لبه ها میتوانند تصویر ورودی را با لبه های مختلف نشان دهند.

در لایه دوم convolutional ، در لایه قبل نورون های مربوط به لبه هایی که تضاد بیشتری با اطرافش دارد زودتر fire میکند . به طور کلی STDP تمایل دارد که آن چیزی که بیشتر تکرار میشود را یاد بگیرد . پس نورون های این لایه ، برجسته ترین ویژگی را یاد میگیرند و پس زمینه هایی که بین تصاویر تغییر میکند ، را یاد نمیگیرد .

در لایه سوم convolutional ، ترکیبی از ویژگی ها تشخیص داده میشود ، مثلا در دیتاست موتور ، چرخ عقب ، دسته، بدنه میانی ... و در دیتاست چهره، نمونه اولیه صورت را به عنوان ترکیبی از ویژگی ها در نظر میگیرد .درواقع این لایه با استفاده از ویژگی هایی که در لایه های قبل شناسایی شده اند ، نمونه های اولیه کل شی را یاد میگیرد. بنابراین STDP صفحه های لایه سوم را جهت یادگیر نمونه ها هدایت میکند . از آنجا که ویژگی ها در تکرارهای اولیه تشکیل نمیشوند، نورون ها به هر الگویی پاسخ میدهند. به همین دلیل اکثر اوقات وزن های سیناپسی از بین میروند و به تدریج به سمت 0.5 و قله های C_1 حرکت میکنند .
با توجه به این که در این جا از $soft - bounded\ STDP$ استفاده کردیم ، learning سریعتر است و زمانی که وزن ها به سمت 0.5 باشند ، C_1 به سرعت کاهش میابد . در پایان learning ، همزمان با شکل گیری ویژگی ها و همگرایی وزن های سیناپسی به 0 یا 1 ، C_1 به 0 میل میکند.
تغییرات وزن در لایه های پایین تر از لایه های بالاتر سریعتر است . و این به این دلیل است که در لایه های بالاتر رقابت برای ویژگی ها است.

در شکل 4 تمام لایه ها و روند گفته شده را میبینیم. و به طور واضح میبینیم که لایه DoG تضادها را مشخص میکند و لایه اول convolutional لبه ها و زوایا را مشخص میکند . لایه دوم convolutional با ترکیب spike های قبلی کیم از ویژگی را تشخیص میدهد و نهایتا در لایه سوم ، کل شی تشخیص داده میشود .
حالا ، اگر داده های تست را امتحان کنیم به دقت 99.1 ± 0.2 میرسیم . و این نشان میدهد که چگونه نمونه های اولیه در لایه های بالاتر تشخیص داده میشوند . اگر بخواهیم با دقت بیشتری بررسی کنیم ، میبینیم که هر نورون در لایه $global\ \max pooling$ تقریبا دقتی برابر با 93% و میانگین دقت 89.8% بود.
نقش قانون STDP : داریم که ، ما ابتدا هر سه لایه convolutional با استفاده از STDP آموزش دادیم تا همگرا شوند . سپس برای لایه convolutional خاص، تعداد سیناپس های فعال از هریک از صفحه ها ویژگی آن را محاسبه میکنیم.

-2 $ETH-80$ dataset :

مجموعه دیتا $ETH-80$ ، شامل 8 دسته مختلف است : سیب ، ماشین ، گاو اسباب بازی ، فنجان ، سگ اسباب بازی ، اسب اسباب بازی ، گلابی و گوجه فرنگی . هر شی از 41 نقطه دید متفاوت ، با زاویه دید متفاوت و شیب متفاوت عکس برداری شده. این دیتاست نمونه خوبی است که نشان دهد SDNN به چه شکل چند شی با تنوع بالا را دسته بندی میکند و چگونه تغییرات بزرگ دید را هندل میکند. ما از SVM خطی برای classifier با پارامتر پنالتی $C = 1.2$ استفاده میکنیم. برای داده های $train$ از هر دسته 5 تا به صورت تصادفی انتخاب میکنیم . ما شبکه ای مانند شبکه دیتاست قبلی داریم. در این جا ما از 400 neuronal map در هر لایه استفاده میکنیم. مانند قبل لایه اول لبه ها را تشخیص میدهد ، لایه دوم ویژگی های میانی و لایه سوم نمونه اولیه را تشخیص میدهد . نورون ها در بالاترین لایه ، به دیدگاه های مختلف اشیا مختلف پاسخ میدهند . پس شبکه برای یادگیری اجسام سه بعدی ، نمونه های اولیه دو بعدی را با مشاهده هر دسته یاد میگیرد . HMAX یکی از مدل های محاسباتی در قشر بینایی است . 5000 ویژگی دارد و از SVM خطی به عنوان $classifier$ استفاده میکند. DCNN با الگوریتم $back\ propagation$ مجدد روی تصاویر با مقیاس خاکستری $ETH-80$ آموزش داده شد . از توابع فعال سازی ReLU و $soft-max$ برای لایه های میانی و تصمیم گیری استفاده شد. hyper parameter های نرخ یادگیری ، $momentume$ ، $regularization$ ، بهینه شده اند . برای جلوگیری از over fitting از $early\ stopping$ استفاده میکنیم. پس DCNN ، میانگی حدود 81.9% دارد.
حال SDNN را با یک DCA مقایسه میکنیم. از تابع فعالیت ReLU در لایه convolutional شبکه DCA استفاده شده . برای آموزش DCA از الگوریتم یادگیری SGD استفاده میکنیم . در زمان همگرایی learning بخش decoder را از DCA حذف میکنیم. و از نمایش طبقه بندی خطی SVM خروجی encoder برای داده های آموزشی و تست استفاده کردیم. مجددا روی تصاویر با مقیاس خاکستری $ETH-80$ آموزش دادیم. DCA دارای دقت 80.7% است. و میبینیم که SDNN از DCA بهتر عمل میکند. حال از SDNN لایه آخر را حذف میکنیم و $global\ \max pooling$ را روی لایه دوم اعمال میکنیم و همچنان از SVM را برای تولید داده آموزشی استفاده میکنیم . میبینیم که دقت 5% کاهش پیدا میکند و برابر با 77.4% میشود . در تحلیل نهایی ، روی SDNN نسبت به داده $ETH-80$ یک ماتریس اعمال میکنیم. میبینیم که بیشتر خطاها به دلیل طبقه بندی نادرست سگ ها ، اسب ها و گاو ها است. میبینیم که خطاها به طور یکنواخت بین نقاط دید مختلف توزیع شده اند . و دلیل خطا هم شباهت کلی بین اشیا است .

-3 MNIST dataset :

این دیتاست دارای 60000 داده آموزشی و 10000 داده تست است . اندازه هر تصویر 28×28 پیکسل است . برای لایه اول مجددا از فیلتر DoG برای $on\ \&\ off\ center$ استفاده میشود. لایه های اول و دوم به ترتیب شامل 30 و 100 صفحه با پنجره های convolutional به اندازه 5×5 و $threshold$ شان برابر با 15 و 10 است. سایز پنجره های $pooling$ لایه اول $pooling$ برابر با 2×2 و $stride = 2$ است. $a^+ = 0.004$ و $a^- = 0.003$. ما مجددا از SVM و $C = 2.4$ استفاده میکنیم. با توجه به شکل 9 ، رنگ های سبز و قرمز با فیلتر های DoG برای

$on\ \&\ off\ center$ تطابق دارد. این لایه تقریبا مانند فیلتر $gabor$ عمل میکند ، یعنی لبه های پیدا شده با جهت گیری ، فاز و قطبیت متفاوت همگرا شده است. این ویژگی های لبه در لایه بعدی ترکیب شده . در این دیتاست ، $unsupervised\ SDNN$ بهتر عمل میکند و ما به دقت 98.4% دست یافتیم.

مهم ترین مزیت SDNN ، فرستادن $spike$ های کم است. SDNN فقط حدود 600 $spike$ برای همه لایه ها استفاده میکند. در حالی که $supervised\ SDNN$ در هر لایه از هزاران $spike$ استفاده میکند . همچنین به دلیل rate − based neural coding نیاز به پردازش در 100 ها مرحله زمانی دارند اما شبکه ما این دیتاست را در 30 مرحله زمانی پردازش میکند.

میدانیم هرگاه یکی از نورون های ما $spike$ بزند ، بقیه نورون های موجود در آن موقعیت را مهار میکند . بنابراین تعداد هر رویداد مهاری در این دیتاست برابر با تعداد $spike$ ها است .