A Deep Understanding of Deep Learning
by Mike X Cohen

فولدر ۳

- در مدل های دیپ لرنینگ از ریاضیات ساده در حد دبیرستان یعنی استفاده از چهارعمل اصلی و تعدادی از توابع ساده مثل مثلثات یا لگاریتم استفاده می شود
- در لایه های شبکه از این توابع و عملیات ریاضی به تعداد بسیار زیاد استفاده می شود و همین باعث امکان انجام محاسبات پیچیده می گردد
- مدل کلی شبکه های عصبی عمیق به این صورت است که ورودی را گرفته و خروجی را پیش بینی می نماید
- برای پیش بینی در کاربردهای مختلف، استفاده از شبکه های عصبی متفاوتی توصیه شده است. مثلا برای پردازش و پیش بینی تصاویر از سی ان ان استفاده می شود و برای پردازش متن یا ترجمه ویا تحلیل سری های زمانی از آر ان ان. برای انجام محاسبات دسته بندی و خوشه بندی از ANN استفاده می شود
- نکته مهم این است که ساختار همه شبکه های عمیق مشابه هم است... مانند ساختمان ها که ممکن است از نظر کاربری و نما و امکانات متفاوت باشند ولی زیر ساخت همه آنها مشابه هم است و از آجر و سیمان و لوله و سیم وغیره تهیه شده اند.
- حرکت رو به جلو به معنی اختصاص وزن های پارامتر های ورودی به معادله می باشد و حرکت برگشت به معنی اخذ بازخورد و نیاز به اصلاح وزن پارامتر ها می باشد. Forward Propagation و Backward Propagation
- مثال حرکت رو به جلو و برگشت: تهیه ساندویچ کره و مربا : تعیین میزان کره و مربا در حرکت رو به جلو و اخذ بازخورد از مشتری در میزان شیرین یا چرب بودن ساندویچ و اصلاح میزان استفاده از آنها در حرکت رو به جلو بعدی
- مشابهت شبکه های عصبی با سلول های عصبی انسان، قیاس درستی نیست و مثل تشابه خودرو و اسب و یا هواپیما با پرنده است. تکنولوژی خصوصیات خود را دارد و قیاس آن با طبیعت درست نیست.
-

فولدر ۵

- تفاوت Complicated , Complex :
  - اشیاء Complicated شامل تعداد زیادی از اجزا ساده هستند که در کنار یکدیگر کار کرده و مجموعه پیچیده ای را بوجود می آورند مثل خودرو یا گوشی موبایل
  - اشیاء Complex دارای تعداد اجزا کمی هستند ولی پیچیدگی آنها زیاد است. مثل علوم پزشکی و Conway’s game of Life
- تفاوت Dummy variables و One-hot encoding :
  - متغیرهای دامی در واقع بصورت صفر و یک وجود یا عدم وجود مورد خاصی را برای یک متغیر مشخص می کنند. مثلا شخص در امتحان قبول یا رد شده است. خروجی یک بردار یا ماتریس یک بعدی است.

- در one-hot encoding چند متغیر داریم که می خواهیم وجود یا عدم وجود آن را بررسی کنیم. در واقع از چند متغیر دامی تشکیل می شود. مثلا شخص در درس ریاضی، تاریخ و علوم نمره قبول یا رد دارد. خروجی یک ماتریس چند بعدی است که از چند ماتریس یک بعدی (دامی) تشکیل شده است.

- تعریف تابع سافت مکس (Softmax):
  - این تابع با محاسبه احتمال هر یک از اعداد ورودی، مجموعه ای از احتمال ها را بعنوان خروجی بر می گرداند. جمع خروجی ها برابر یک می باشد.
  - فرمول محاسبه احتمال برای هر i برابر است با : $\frac{e^{z_i}}{\sum e^z}$
  - بزرگترین عدد در این تابع بیشترین مقدار احتمال و کوچکترین آنها، کمترین احتمال را دارد.

- آنتروپی و کراس آنتروپی
  - به‌طور کلی در علوم و مهندسی، آنتروپی معیاری از میزان ابهام یا بی‌نظمی است.
  - آنتروپی به معنای این است که هر چه میزان عدم قطعیت یا پیش بینی پذیر بودن مقادیر بالا تر یا پایین تر باشد، مقدار آنتروپی به صفر نزدیکتر می شود. یعنی هرچه احتمال رخداد اتفاقی بیشتر باشد، آنتروپی آن کمتر است. از طرفی هر چقدر احتمال رخ دادن اتفاقی کمتر باشد، آنتروپی آن کمتر است. مثلا در پرتاب سکه، احتمال آنکه از ۱۰۰ پرتاب هر صد مرتبه شیر بیاید نزدیک به صفر است و آنتروپی آن هم صفر است. آنتروپی احتمال آنکه ۵۰ درصد شیر بیاید یک است. هر چقدر احتمال شیر آمدن سکه به ۰/۵ نزدیکتر باشد ابهام در مورد نتیجه آن بیشتر است و اطلاع از نتیجه، به‌طور میانگین اطلاعات بیشتری دربردارد.

$$H(p) = -\sum p(x)\log(p(x))$$

  - برای شرایطی که دو خروجی متصور هستیم مثل تشخیص تصویر گربه که در واقع خروجی درست یا غلط هست فرمول آنتروپی به شرح زیر است:

$$H(p) = -(p\log(p) + (1-p)\log(1-p))$$

اگر احتمال p= 0.25 باشد، آنگاه آنتروپی یعنی ابهام در تشخیص نتیجه حدود 0.56 می باشد.

  - کراس آنتروپی احتمال رخ داد یک اتفاق به شرط اتفاق دیگر را بررسی می کند. در واقع در این زمینه دوتوزیع احتمال دخیل هستند.

$$H(p) = -\sum p(x)\log(q(x))$$

  - در مثال تشخیص گربه اگر دسته بندی گربه بصورت صفر و یک باشد و احتمال دیده شدن گربه 0.25 و دیده نشدن 0.75 باشد، آنگاه فرمول محاسبه کراس آنتروپی باینری (صفر و یک) به شرح زیر است.

$$H(p) = -(1 * \log(0.25) + (0)\log(0.75))$$

- گرادیان کاهشی (Gradient Descent) : برای پیدا کردن نقطه مینیموم تابع می توانیم از مشتق تابع استفاده کنیم و در نقطه ای که شیب (مشتق) برابر صفر می شود (نقطه ماکزیموم یا مینیموم) با توجه به مقدار مشتق در قبل و بعد از نقطه مشخص شده تشخیص دهیم که مینیموم یا ماکزیموم است. در نقطه مینیموم مقدار مشتق یا شیب خط قبل از نقطه منفی و بعد از آن مثبت می شود.

- گرادیان محو شونده (Vanishing Gradient) : در مواقعی ممکن است مقدار شیب خط یا مشتق تابع بدون اینکه در نقطه مینیموم یا ماکزیموم باشد برابر با صفر شود. این نقاط جایی هستند که مقدار تابع ثابت (Constant) می شود و شیب یا مشتق آن تغییر نمی کند و برابر صفر می شود. به این حالت گرادیان محو شونده (Vanish) گویند.

فولدر ۶ : گرادیان کاهشی

یکی از مهمترین الگوریتم های شبکه های عصبی و یادگیری عمیق الگوریتم گرادیان کاهشی است و بدون آن بی معنی هستند

- چگونگی یادگیری عمیق توسط مدل:
  - تخمین پارامترها
  - محاسبه خطا از پارامترهای تخمینی
  - یادگیری از خطاها و اصلاح پارامترها (گرادیان کاهشی)
- در واقع در یادگیری عمیق ابتدا وزن هر متغیر بصورت رندوم انتخاب می شود و سپس بر اساس آنها مقادیر محاسبه و نیز خطا محاسبه می گردد. سپس بر اساس میزان خطا و اختلاف آن با مقادیر واقعی، سعی می کنیم مینموم خطا را بدست آوریم که در اینجا از گرادیان کاهشی (مشتق تابع) استفاه می کنیم.
- روش کار به این صورت است که در یک نقطه اتفاقی (رندوم) مقدار تابع و خطا محاسبه می شود. اگر مقدار خطا زیاد باشد، نقطه انتخاب شده باید اصلاح شود تا به نقطه مینیموم تابع خطا برسیم. برای اصلاح مقدار **شیب نقطه ضربدر نرخ یادگیری (Learning Rate) را محاسبه** و از نقطه کم می کنیم. بدین صورت به نقطه مینیموم نزدیک می شویم.
- زمانیکه مقدار محاسبه شده بالا مساوی صفر و یا خیلی نزدیک به صفر باشد، گرادیان محو شونده (Vanishing) اتفاق می افتد.
- نکته مهم اینکه گرادیان کاهشی ممکن است دقیقا روی نقطه مینیموم قرار نگیرد. دلیل این امر این است که میزان حرکت از نقطه انتخابی اول با نرخ یادگیری مشخص می شود و ممکن است جایی در اطراف و نزدیک به نقطه مینیموم قرار گیرد. لذا انتخاب نرخ یادگیری در مدل اهمیت ویژه ای دارد.
- در انتخاب مقدار مینیموم و بدست آوردن بهترین نقطه، ممکن است مینیموم های محلی(Local) باعث ایجاد خطا در بدست آوردن مینیموم کلی (Global) شوند. به دلیل زیر در یادگیری عمیق این مورد اهمیت زیادی ندارد:
  - در شبکه های یادگیری عمیق معمولا تعداد متغیرها بسیار زیاد است و لذا معادلات دارای ابعاد (بعدها) زیادی هستند. این موضوع باعث می شود که نقطه مینیموم متغیرها در یک نقطه محلی اتفاق نیافتد و جایی که یک متغیر در مینیموم قرار گرفته است سایر متغیرها لزوما در مینیموم نباشند. مثلا نموداری که مشابه زین اسب باشد را در نظر بگیریم. در یکی از بعدها یعنی یک متغیر در مینیموم است ولی در متغیرهای دیگر در ماکزیموم قرار گرفته است.
  - بطور خلاصه می توان گفت که در شبکه های یادگیری عمیق اتفاق افتادن مینیموم محلی تقریبا ناممکن است.

- زمانی که عملکرد مدل خوب است، می توان با انجام کارهای زیر از درست بودن عملکرد و اطمینان از انتخاب نقطه مینیموم:
  - چندین بار مدل را آموزش دهیم و نتیجه را بررسی نماییم تا مطمئن شویم که عملکرد مدل درست است. در واقع در هر بار شروع چون از نقطه جدیدی بصورت اتفاقی کار آغاز می گردد، می تواند نقطه مینیموم را با دقت متفاوتی تخمین بزند.
  - ابعاد مساله را زیاد کنیم تا مطمئن شویم که نقطه مینیموم محلی وجود ندارد.
- **گرادیان** مجموعه ای از دو یا چند مشتق تابع از متغیرهای مختلف است. اگر متغیرهای معادله x, y, z باشند و از معادله نسبت به تک تک آنها مشتق گرفته شود و در یک لیست قرار گیرد، به این لیست گرادیان گفته می شود.

- در محاسبه گرادیان کاهشی، با تغییر نرخ یادگیری و نیز تعداد مراحل اجرای محاسبه مشتق و محاسبه نقطه جدید (epoch) می توان به جواب بهینه و نقطه مینیموم کلی رسید. البته اگر مدل دچار نقطه مینیموم محلی نشود (مدل هایی با تعداد متغیرهای کم).
- نرخ یادگیری می تواند بر اساس زمان (epoch) ، بر اساس شیب خط (مشتق تابع) متغیر باشد و یا بصورت ثابت در نظر گرفته شود.

گرادیان انفجاری (Exploding Gradient):

زمانی که شیب تابع (مشتق) خیلی زیاد شود، زمانیکه حالت عمودی به محور ایکس داشته باشد، مقدار محاسبه شده میزان تغییر شیب ضربدر نرخ تغییرات بسیار بزرگ می شود و وقتی این عدد را از نقطه ای که در آن قرار داریم کم یا زیاد کنیم، ممکن است از روی نقطه بهینه مینیموم پرش کنیم و آن نقطه را نبینیم. در این حالت گرادیان انفجاری رخ داده است.

فولدر ۷ : Artificial Neural Network (ANN)

- مفهوم پرسپترون (Perceptron): متغیرهای ورودی شامل مقدار ثابت (Bias) و نیز مغیرهای مستقل بهمراه وزن مربوطه که مانند یک ضریب برای آنها عمل می کند وارد یک تابع می شوند که آنها را بصورت خطی با هم جمع می کند. سپس وارد یک تابع غیر خطی می شوند که تابع فعال ساز (Activation Function) می باشد. و در نهایت یک پیش بینی متغیر هدف (وای هت) بعنوان خروجی خواهیم داشت. به مجموعه تابع خطی و تابع غیر خطی فعال ساز پرسپترون گفته می شود.

- مفهوم تابع فعال ساز (Activation Function): تابعی غیر خطی است که بر اساس ورودی هایی که به آن داده می شود، خروجی متفاوتی را بر می گرداند. معروف ترین و پرکاربردترین توابع فعال ساز، Relu ، Sigmoid و Hyperbolic tangent می باشند.

  - تابع Relu : برای مقادیر کمتر از صفر خروجی صفر دارد و برای مقادیر بیشتر از صفر همان مقدار را بر می گرداند. از این تابع معمولا در لایه های میانی استفاده می شود.
  - تابع Sigmoid : برای مقادیر منفی تا صفر خروجی ۰ تا ۰,۵ می دهد و برای مقادیر بیشتر از صفر از ۰,۵ تا یک خروجی می دهد. از این تابع معمولا در لایه آخر (خروجی) استفاه می شود. در واقع برای دسته بندی باینری بسیار مناسب است و نتایج بیشتر از صفر در دسته یک و نتایج کمتر از صفر در دسته صفر قرار می گیرند.
  - تابع Hyperbolic tangent : برای مقادیر منفی تا صفر مقدار -۱ تا صفر بر می گرداند و برای مقادیر مثبت از صفر تا مثبت یک بر می گرداند.

- مفهوم تابع خطا (Loss Function) : MSE جهت پیش بینی متغیرهای پیوسته مثل قیمت مسکن، دمای هوا و قد نفرات است استفاده می شود و Cross-Entropy (logistic) جهت متغیرهای گسسته مثل رد یا قبول شدن در امتحان، وجود یا عدم وجود تصویر حیوان در عکس، تحلیل احساسات منفی یا مثبت استفاده می شود.
    - MSE با فرمول زیر برای مقایسه متغیر هدف اصلی و متغیر هدف پیش بینی شده بکار می رود.
    - $L = \frac{1}{2}(\hat{y} - y)^2$

    - در محاسبه کراس آنتروپی احتمال رخ دادن یک پدیده (پیش بینی متغیر هدف) را با رخ دادن پدیده (متغیر هدف واقعی) مقایسه می کنیم. مقدار منفی در فرمول به دلیل لگاریتم مقادیر کمتر از یک می باشد که باعث ایجاد نتیجه منفی در داخل پرانتز می کند.

    - $L = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$

- مفهوم تابع هزینه (Cost function): وقتی تابع خطای متغیرها را با هم جمع و تقسیم بر تعداد کنیم یعنی میانگین توابع خطا را محاسبه کنیم به تابع هزینه دست پیدا می کنیم. در واقع تمام متغیرهای هدف پیش بینی شده را با یکی از روش های فوق با مقدار متغیر هدف واقعی مقایسه و تابع خطا را بدست می آوریم و میانگین آنها را بدست می آوریم.

$$J = \frac{1}{n}\sum L\,(\,y_i\ .\ \hat{y}_i)$$

- هدف از محاسبه توابع هزینه و خطا محاسبه وزن های هر متغیر و در نتیجه کاهش حداکثری این توابع (هزینه و خطا) می باشد.
- استفاده از تابع خطا برای بهینه سازی ممکن است باعث ایجاد حجم زیاد محاسبات و نیز ایجاد اورفیت در مدل شود. لذا از تابع هزینه جهت انجام محاسبات استفاده می نماییم و این کار را بصورت بچ به بچ انجام می دهیم.
- به مجموعه ضرایب مربوط به ورودی ها و نیز جمع خطی آنها (سیگما) و اعمال تابع فعال سازی (اکتیویشن) یک نود (node) گفته می شود.
- هر نود ممکن است چندین **ورودی** داشته باشد ولی **فقط یک خروجی** دارد که برای ورود به نودهای دیگر در لایه های بعدی به تعداد مورد نیاز کپی می شود.
- هر نود بصورت کاملا مجزا از بقیه کار می کند و کاملا منحصر بفرد است. در واقع کارکرد نود ها روی همدیگر اثری ندارد و فقط ورودی ها هستند که بر خروجی نود اثر گذار هستند.

مفهوم حرکت برگشت (Back Propagation)

- زمانیکه ضرایب متغیرها بصورت اتفاقی انتخاب و بر اساس آن پیش بینی ارائه می گردد، نیاز است تا به سمت بهترین و کمترین خطای ممکن حرکت کنیم. این تغییر بصورت گرادیان کاهشی اتفاق می افتد که از نظر مفهومی با حرکت برگشتی معادل است. در واقع حرکت ما به عقب بمنظور اصلاح **ضرایب متغیرها** بوسیله روش گرادیان کاهشی را حرکت برگشت گویند.

- یادآوری اینکه در گرادیان کاهشی مقدار متغیر به اندازه نرخ تغییر (شیب یا مشتق) ضربدر نرخ یادگیری کاهش یا افزایش می یافت. در اینجا هم **میزان تغییر ضرایب** به اندازه شیب یا مشتق تابع نسبت به ضریب ضربدر نرخ یادگیری تغییر (افزایش یا کاهش) می یابد.
- مقدار مشتق یا شیب تابع نسبت به ضرایب به **نوع تابع خطا** (یا تابع هزینه) و **تابع فعال سازی** بستگی دارد.

حل مساله رگرسیون خطی بوسیله یادگیری عمیق

- در این مسایل چون از تابع فعال سازی غیر خطی استفاده می شود، ممکن است نتایج بصورت یک خط کامل بدست نیاید و در بعضی نقاط غیر خطی دیده شود.

حل مساله دسته بندی باینری با ANN

- در حل این مسائل باید دقت شود که لزوما بهترین نتیجه از روش های یادگیری عمیق بدست نمی آید و شاید استفاده از روش های یادگیری ماشین مثل KNN یا Kmeans نتیجه بهتری داشته باشند.
- تغییر در نرخ یادگیری می تواند نتایج بسیار متفاوتی را در مدل ایجاد کند. البته با توجه به ماهیت اختصاص اتفاقی ضرایب به متغیرها، ممکن است تغییرات نرخ یادگیری در یک بار آموزش و اجرای کد نتیجه ندهد و لازم باشد که بارها تکرار شود تا بهترین نتیجه بدست آید.
- با تغییر نرخ تغییر در مدل، دقت (accuracy) از ۵۰ درصد تا حدود ۹۹ درصد تغییر می کند.
- استفاده از توابع غیر خطی فعال ساز در مسایل خطی و بطور کلی استفاده از روش های پیچیده در حل مسائل ساده باعث عدم بازدهی مدل های می گردد.
- بطور خلاصه برای مسائل ساده باید از روش های ساده استفاده کرد.
- با حذف توابع غیر خطی فعال ساز، دقت مدل از ۹۹,۵ تا ۱۰۰ درصد ارتقا می یابد. این در حالیست که کلیه پارامتر های مدل و نرخ یادگیری و ... همانند سابق باقی می مانند.

استفاده از لایه های داخلی (پنهان)

- در پردازش داده ها در شبکه های عصبی می توانیم لایه ها را بصورت جفت خطی و غیر خطی (تابع فعال سازی) در نظر بگیریم و به مدل اضافه نماییم.
- تعداد ورودی مدل به تعداد خصوصیات (فیچرها) بستگی دارد و تعداد لایه های داخلی به دلخواه انتخاب می شود. خروجی لایه اول بعنوان ورودی لایه بعدی می باشد و باید تعداد ورودی لایه بعدی با خروجی لایه قبلی یکسان باشد. خروجی لایه آخر با توجه به نوع مساله مشخص می شود.
- معمولا در لایه های میانی از تابع فعال ساز Relu استفاده می شود و در لایه آخر و خروجی نهایی مدل از Sigmoid استفاده می کنیم.
- اگر در مسائل دسته بندی از تابع سیگموید استفاده گردد، تابع خطا BCELoss در نظر گرفته می شود.
- می توانیم بجای استفاده از BCELoss از تابع BCELogitsLoss استفاده کنیم و در این حالت باید تابع سیگموید را از مدل حذف نماییم.

نکته مهم:

در حل مسائل غیر خطی (Non-Linear) پیچیده از مدل های یادگیری عمیق استفاده می نماییم. اینکه می توانیم و دانش ایجاد مدل های یادگیری عمیق را داریم به این معنی نیست که همه مسائل را در این چارچوب و مدل ها حل کنیم.

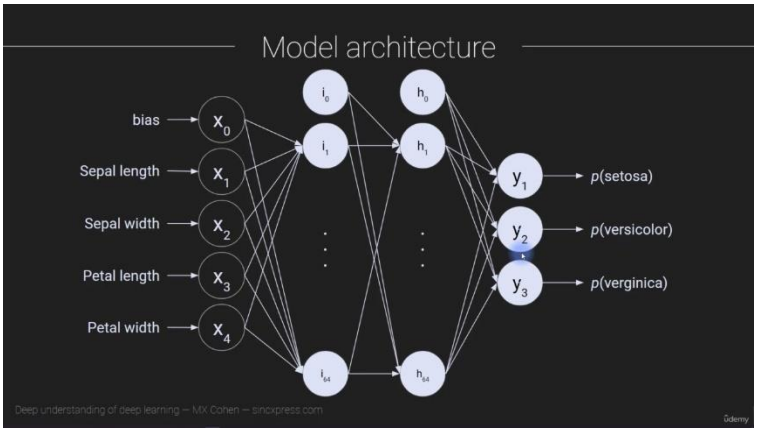
Just because you can, doesn’t mean you should.

برای حل هر مساله بهتر است بهترین و خلاقانه ترین و در عین حال ساده ترین مدل ها را در نظر بگیریم.

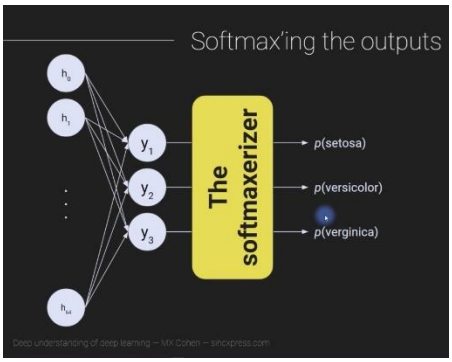
- مدل های حل مسائل خطی فقط یک لایه دارند. در واقع به دلیل ماهیت الگوهای معادلات خطی، لایه های میانی در عملیات ریاضی حذف خود به خود می شوند.

حل مساله دسته بندی گل های Iris

- در این مساله از روش یادگیری عمیق استفاده می کنیم. چهار خصوصیت (فیچر) داریم و دسته بندی نهایی در سه دسته خواهد بود.
- نکته در مورد bias: مقدار بایاس در تمام لایه ها برابر یک می باشد و ورودی از لایه قبلی ندارد و تنها به لایه بعدی وارد می شود.



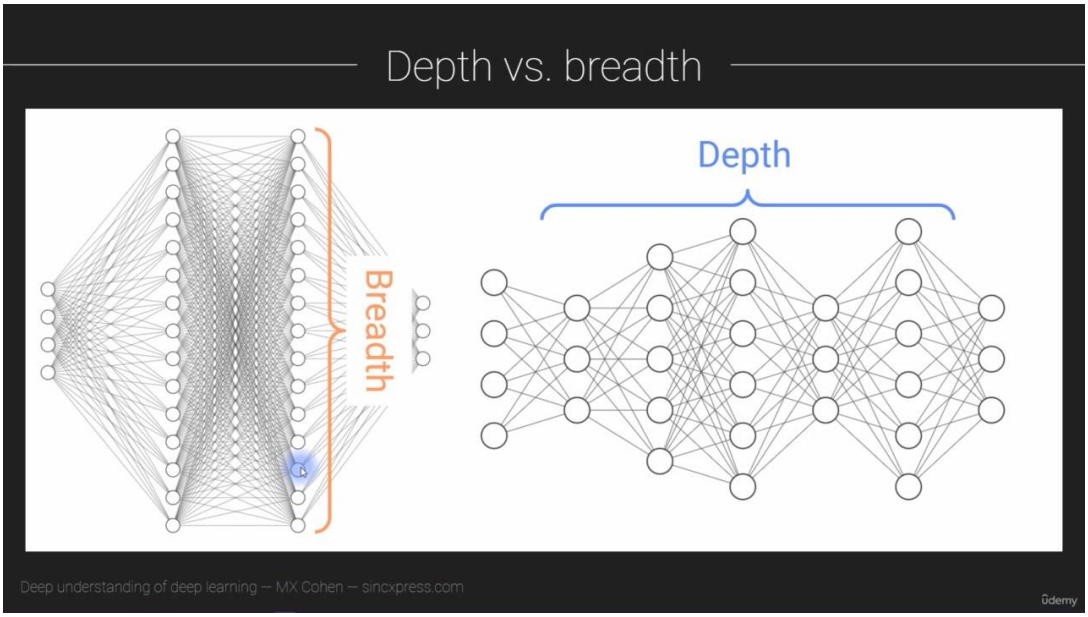
- در خروجی این مدل که سه دسته از گل ها باید مشخص شوند، نمی توانیم از تابع فعال سازی سیگموید استفاده کنیم چراکه در این تابع فقط دو خروج بصورت صفر و یک خواهیم داشت.
- در این مدل از سافت مکس (Softmax) استفاده می شود که خروجی این تابع مقدار احتمال رخ داد می باشد.
  - فرمول محاسبه احتمال در سافت مکس برای هر i برابر است با : $\frac{e^{z_i}}{\sum e^{z}}$
-



- در واقع سافت مکس بیشترین احتمال را به بیشترین مقدار ورودی می دهد.

پهنا و عمق مدل یادگیری عمیق (Depth and Breadth or width)

- تعدا نود ها در یک لایه را پهنا (Breadth or width) و تعداد لایه ها را عمق (Depth) مدل گویند.

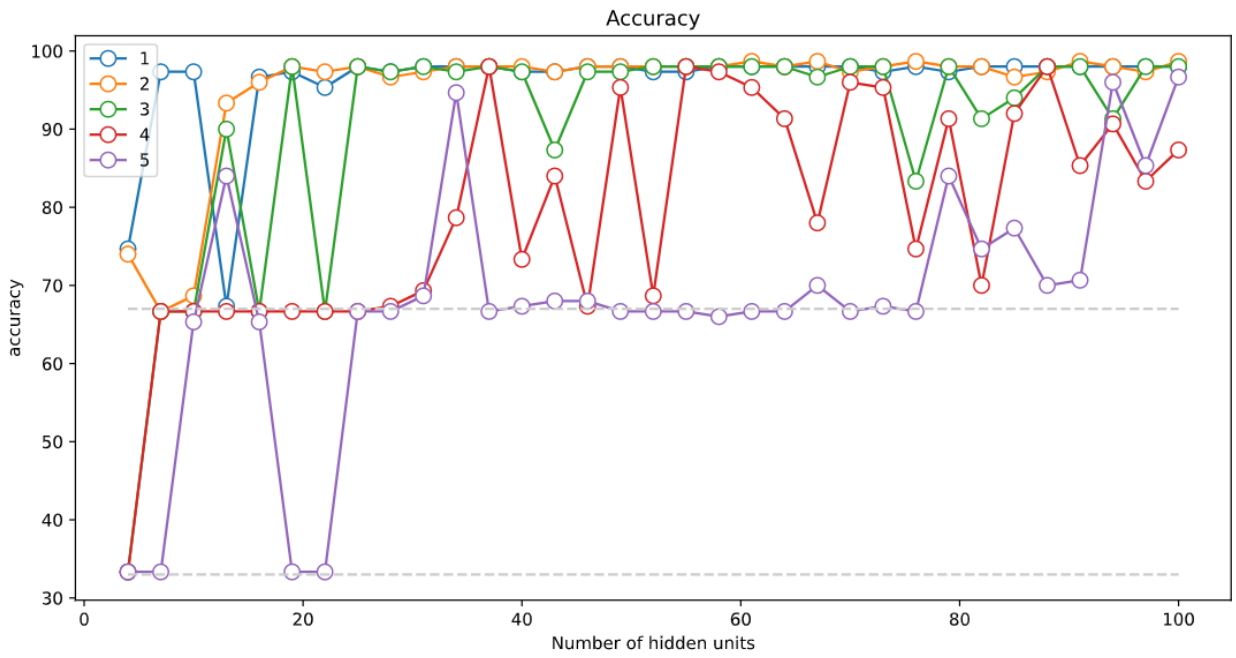


- تعدا پارامترهای یادگیری (تعداد فلش ها که نشان دهنده وزن متغیرها هستند) در مدل هایی با تعداد نود های یکسان که در پهنا و عمق با هم متفاوت هستند تفاوت می کند.

See the following code file

ANN_01.ipynb

- نکته اینکه تعداد لایه های عمق یا پهنا لزوما باعث ایجاد شرایط دقیق تر نخواهند شد. در واقع افزایش تعداد نودها در لایه یا افزایش تعداد لایه ها لزوما باعث بهتر شدن کارآیی مدل نمی شوند.
- نمودار زیر نشان می دهد که افزایش تعداد نود ها در لایه تاثیر چندانی در سرعت یادگیری یا عملکرد مدل و دقت آن ندارد. ضمن اینکه تعداد لایه ها که با رنگ های مختلف در نمودار مشخص شده است نشان می دهند که مدل با تعداد لایه های کمتر می تواند سریعتر به دقت بالاتر برسد.



- بطور خلاصه می توان گفت مدل هایی با عمق کم سرعت یادگیری بیشتری دارند و مدل هایی با عمق بیشتر می توانند موارد و مسائل پیچیده تر را حل کنند.
- کارآیی مدل لزوما به تعداد پارامترهای یادگیرنده آن (وزن ها یا فلش ها) ربطی ندارد.

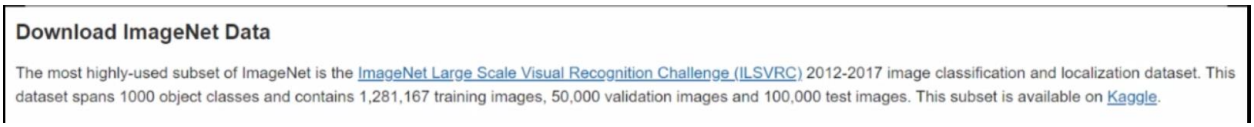
فولدر ۸ : اورفیت و کراس ولیدیشن (Overfitting and Cross-Validation)

- تعداد پارامترهای مدل را چگونه تعیین کنیم؟
  - اگر مساله دارای یک یا دو پارامتر باشد، می توان با مشاهده داده ها در نمودار روابط ریاضی بین آنها را بطور چشمی تقریب زد.
  - اگر مساله دارای ابعاد و پارامتر های بیشتری باشد، روش چشمی و ترسیم نمودار مفید و قابل استفاده نخواهد بود. در این مسائل از روش های آماری برای تعیین تعداد پارامتر ها استفاده می کنیم. یکی از این روش ها **کراس ولیدیشن** است

- اورفیت امکان تعمیم دادن مدل بر روی داده های جدید را ضعیف می کند ولی اگر بخواهیم از همان داده ها و همان شرایط استفاده کنیم، مدل بهتری خواهد بود. در واقع اگر نخواهیم که داده های جدید به مساله بدهیم و پیش بینی بگیریم بحث اورفیت مشکلی ایجاد نمی کند.

- تعمیم (Generalization) و حدود آن (Generalization Boundaries) در بحث اورفیت بسیار مهم هستند. بطور مثال اگر بخواهیم قیمت مسکن در شهری را ارزیابی و مدل آن را تهیه نماییم شاید نتوانیم با همان مدل قیمت مسکن در شهر دیگری را ارزیابی یا پیش بینی نماییم. لذا حدود مدل ما محدود به شهر اول می باشد.

- دیتاست مربوط به imagenet



- استفاده از تابع train-test-split در کتابخانه سایکیت لرن و سپس بکارگیر مدل ANN روی آن جهت داده های گل های Iris

فولدر ۹ : Regularization

فولدر ۱۰ : ۱۰ Metaparameters (activations, optimizers)

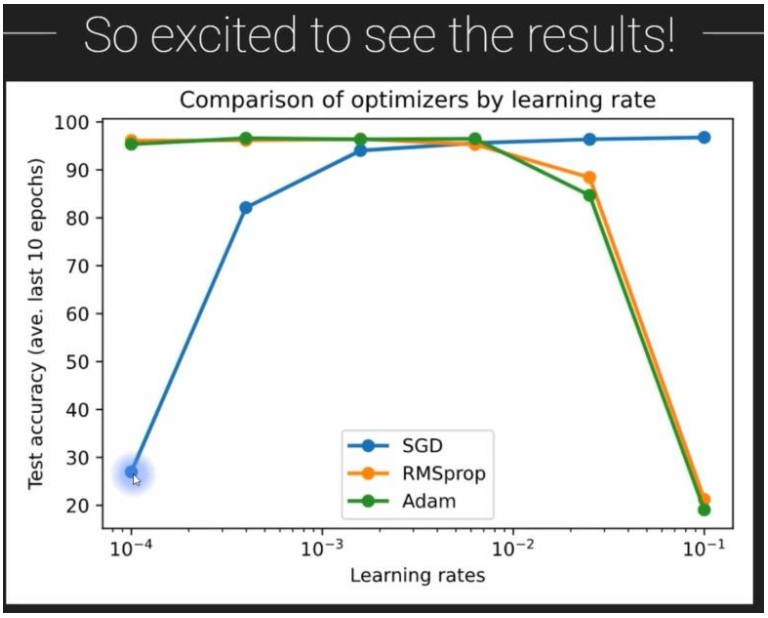
فولدر ۱۱ : FFN

دیتاست MNIST

file : MNIST_1400_12_04.ipynb

- مقایسه انواع بهینه ساز ها (Optimizers) با اختصاص نرخ یادگیری های متفاوت

همانطور که در نمودار دیده می شود، استفاده از نرخ یادگیری کمتر در گرادیان کاهشی اثر منفی دارد ولی در دو بهینه ساز دیگر اثر خوبی گذاشته است. دلیل این موضوع این است که در بهینه ساز آدام و آر ام اس پراپ روش دینامیک برای تغییر نرخ یادگیری پیش بینی و تعبیه شده است.



فولدر ۱۲ : More on Data

در این بخش استفاده از انواع دیتاست ها و نیز استفاده از داده های دیتاست های آنلاین و وارد کردن اطلاعات به گوگل کولب و طریقه استفاده از آنها بررسی می شود.

- در فرآیند استفاده از داده ها در پایتورچ، ابتدا باید داده ها را از قالب نامپای به تنسور تبدیل کرد. سپس باید داده ها و لیبل ها را بوسیله دستور TensorDataset به دیتاست تبدیل کرد.
- استفاده از دیتا در ورودی مدل های شبکه های عصبی اگر بصورت بچ به بچ صورت پذیرد باعث افزایش چشمگیر راندمان و کارکرد این شبکه ها می گردد لذا از دستور dataloader برای این کار یعنی وارد کردن داده ها بصورت بچ به بچ و قسمت بندی شده استفاده می کنیم.
- دیتالودر داده ها را دسته بندی می کند ولی بصورت یک شی قابل استفاده در چرخش ها (Iterable) قابل استفاده است و در آن صورت می تواند داده ها را به مدل وارد نماید.

- داده هایی که تعداد آنها کم است و برای آموزش مدل کفایت نمی کند، و یا نوع داده ها همخوانی ندارند مثل جداسازی تصاویر گربه از قایق نیاز به انجام کار اضافی روی داده ها داریم. در این شرایط می توان از روش های زیر استفاده نمود:
  - داده های بیشتری بدست آوریم..... گاهی اوقات شدنی نیست.
  - تعدادی از داده های بیربط را حذف کنیم.....زمانیکه داده های زیادی داریم که داده های پرت در آنها وجود دارد
  - از داده های موجود بارها و بارها بصورت تکراری نمونه برداری نماییم..... احتمال اورفیت شدن را زیاد می کند.
  - تقویت داده ها (Data Augmentation) به این معنی که خصوصیات غیر خطی به داده ها اضافه نماییم....مثل لگاریتم گرفتن یا پردازش هایی روی تصاویر بعنوان داده های ورودی
  - تولید داده های مشابه داده های فعلی....

SMOTE: Synthetic Minority Oversampling Technique

  - گاهی اوقات ممکن است بتوانیم از روشی غیر از شبکه های عصبی و یادگیری عمیق، مساله را حل کنیم. داده های نامتوازن در شبکه های عصبی می تواند مخرب باشد و لذا شاید بهتر باشد از انواع دیگر مدل های ماشین لرنینگ استفاده کنیم.
- در تقویت داده ها (Data Augmentation) باید دقت کنیم که لزوما با اضافه کردن یک خصوصیت ترکیبی از داده ها، خصوصا اگر این خصوصیت بصورت خطی اضافه شود، ممکن است در کارکرد مدل تاثیر گذار نباشد. اگر خصوصیت ترکیبی غیر خطی اضافه شود ممکن است شرایط بهتر باشد ولی لزوما به معنی رفع مشکلات قبلی نیست.

فولدر ۱۵: Weight inits and investigations

- ماتریس های وزن متغیرها در مدل های شبکه های عصبی بر مبنای تعداد ورودی و خروجی ها مشخص می شوند. با توجه به نوع معادلات جبری که در این شبکه ها نوشته می شود و نیاز به انجام ضرب ماتریسی متغیر ها و ضرایب، در صورت فراخوان کردن ماتریس ضرایب در مدل (پای تورچ) ابتدا تعداد خروجی (پارامتر اول لیست یا تنسور) و سپس تعداد ورودی (پارامتر دوم لیست یا تنسور) نمایش داده می شود. مثلا اگر مدل دارای ۳ ورودی و لایه میانی با ۵ و ۸ نود و خروجی ۴ تایی باشد، در صورت فراخوان ماتریس ضرایب (weights) خروجی بصورت زیر نمایش داده می شود:

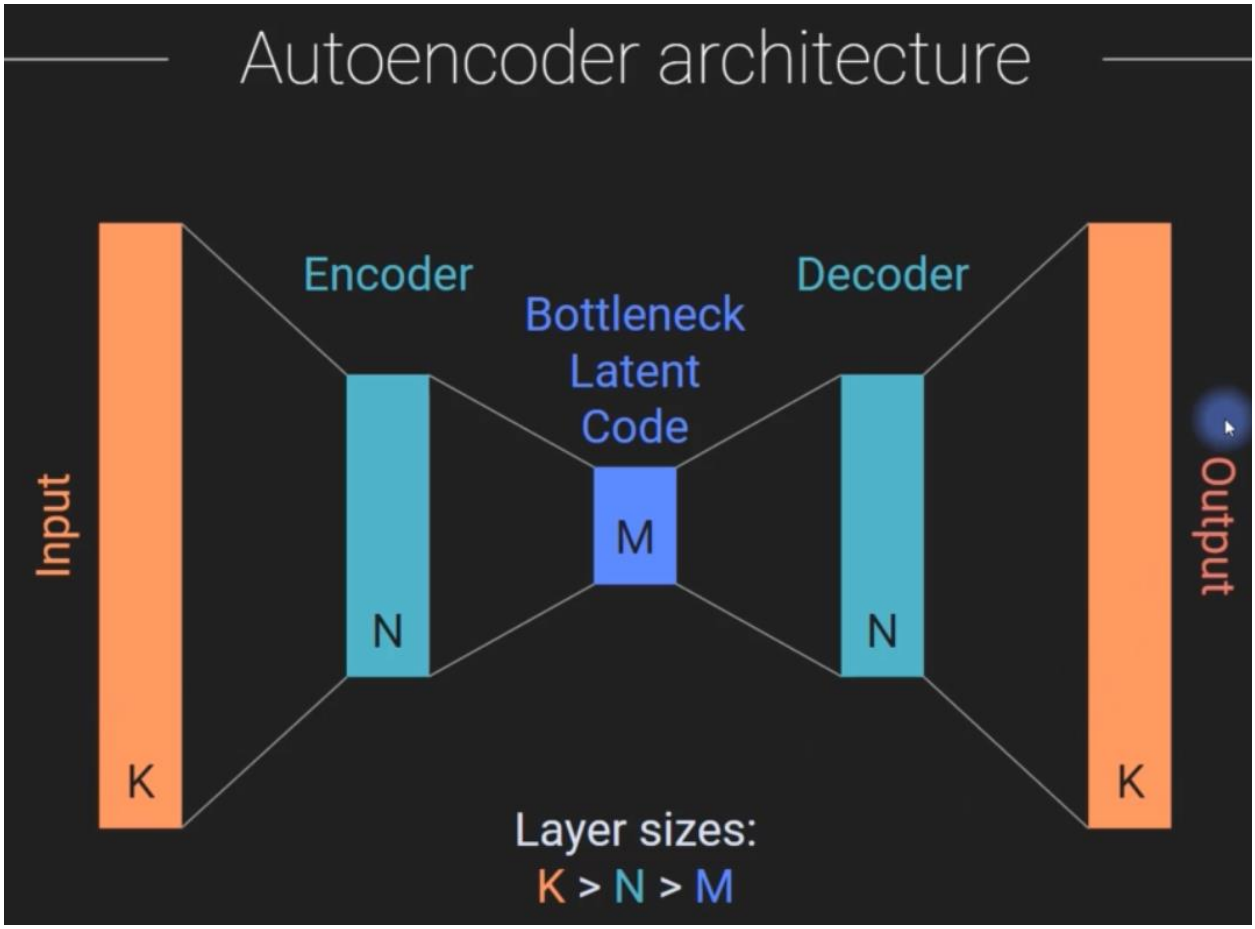
(5, 3) , (8, 5), (4, 8)

- اختصاص وزن اولیه به مدل :
  - در صورتیکه وزن های اولیه ای به مدل اختصاص دهیم، باید دقت کرد که اعداد مختلف بصورت رندوم باشد. در حالتیکه از اعداد یکسان استفاده شود، هیچ مدل یادگیری عمیقی نمی تواند درست عمل نماید.
  - در مسائل نسبتا ساده اختصاص وزن اولیه در مدل یادگیری عمیق کمک زیادی به کارکرد مدل نمی کند ولی در مسائل بزرگ با داده های بسیار زیاد بهتر است این کار انجام شود.

فولدر ۱۶ : Autoencoders

- منظور از اتوانکودر سیستمی است که بتواند خود را برای انکود کردن (معمولا ساده سازی یا کاهش بُعد) داده ها آموزش دهد.

- ساختار اتوانکودر به صورت زیر است:



فولدر ۱۷ : Running models on GPU

- برای استفاده از جی پی یو، علاوه بر استفاده از سخت افزار باید در کدهای برنامه و نرم افزار نیز استفاده از آن قید گردد.
- در پای تورچ از دستورات زیر برای بکار گیری جی پی یو استفاده می شود.

```
-   # use GPU
-   device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

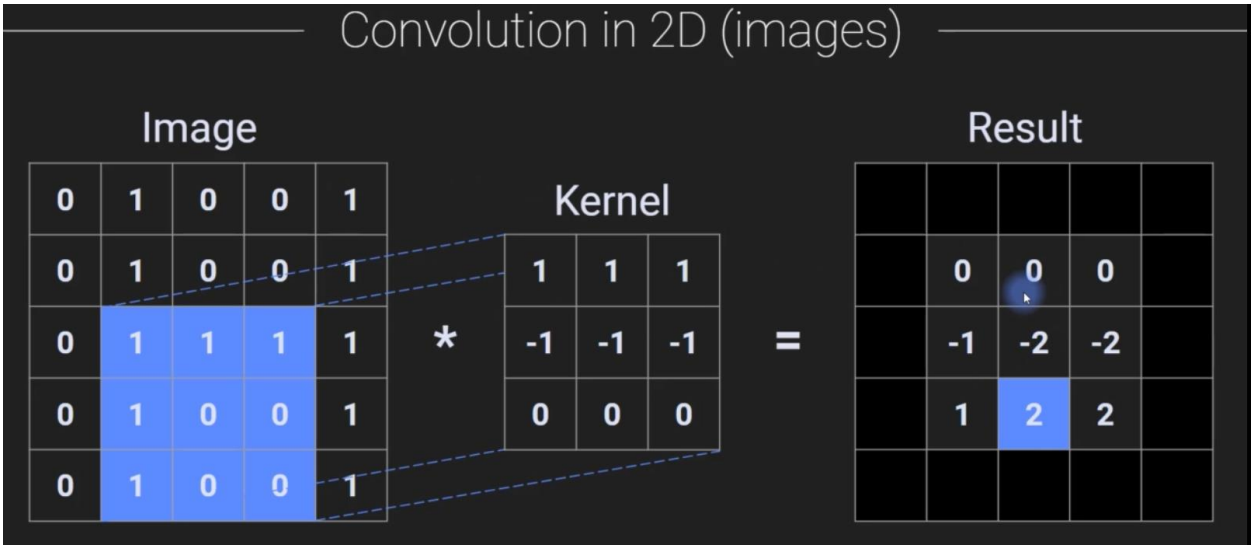
در واقع زمانی که cuda که نشانگر امکان استفاده از جی پی یو است درست باشد از جی پی یو استفاده می کند و در غیر اینصورت از سی پی یو استفاده می شود.

- در برنامه های کوچک و کم حجم استفاده از جی پی یو ارجحیت چندانی ایجاد نمی کند.
- بهتر است از جی پی یو سایت هایی مثل گوگل کولب و یا AWS یا AZURE استفاده نماییم که در حال حاضر یکی از بهترین ها گوگل کولب است.

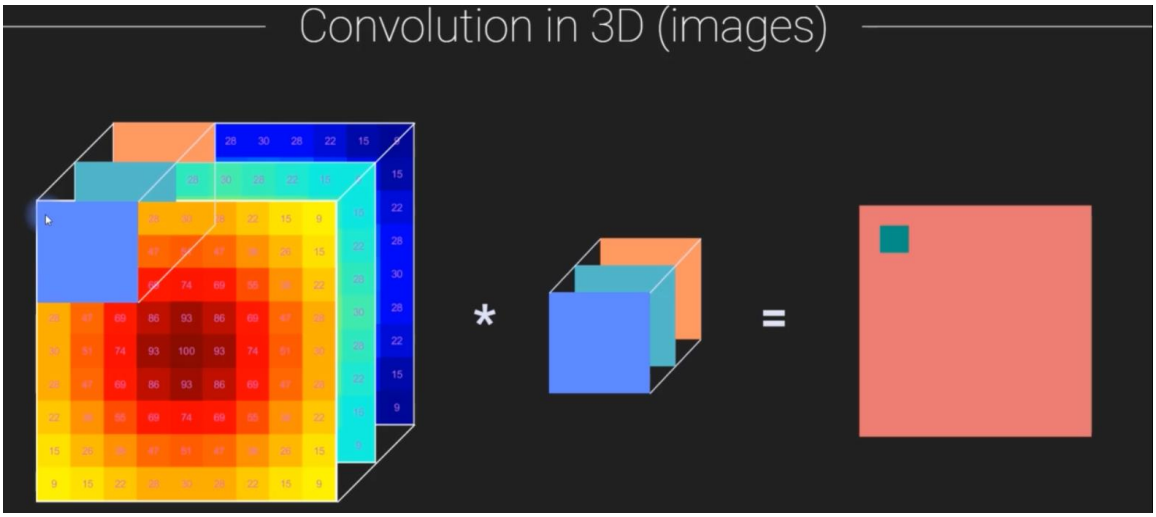
- در صورتیکه زیاد و نابجا از جی پی یو گوگل کولب استفاده نماییم ممکن است برای ساعاتی یا چند روزی دسترسی را محدود سازد ولی دوباره برقرار می شود.
- در حین انجام فرآیند یادگیری مدل، هرجا که لازم باشد می توانیم با وارد کردن آرگومان cpu در پارامتر device در دستورات پای تورچ بجای جی پی یو از سی پی یو استفاده نماییم.

فولدر ۱۸ : Convolution and transformations

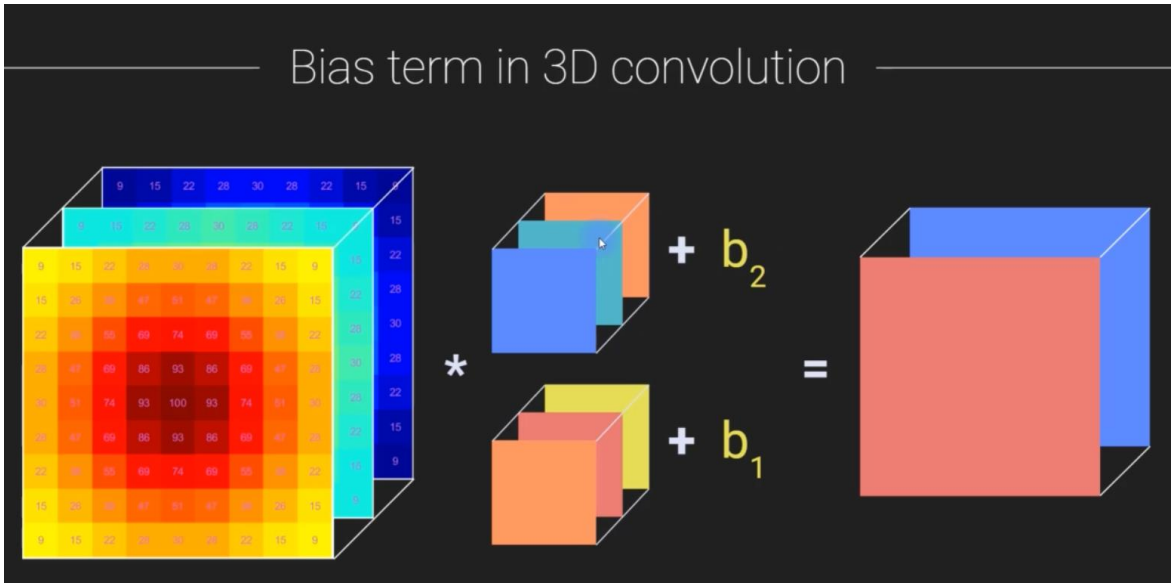
- منظور از کانولوشن استفاده از یک ماتریس ضرایب به نام کرنل (kernel) و ضرب ماتریسی (دات پروداکت) آن در ماتریس داده ها می باشد.
- برای آنکه عملیات ضرب ماتریسی بدرستی صورت پذیرد می توان کناره های ماتریس بزرگتر را با اعداد صفر اضافه کرد (padding or zero padding) تا ماتریس حاصلضرب نهایی با ابعاد مشابه ماتریس اصلی بدست آید.
- در تصویر زیر پدینگ صورت نگرفته است و لذا ماتریس حاصلضرب به اندازه ماتریس کرنل است.



- ماتریس های تصاویر رنگی که حاوی لایه های رنگ قرمز و سبز و آبی هستند با سه ماتریس متناظر کرنل ضرب و به آن کانولوشن سه بعدی گویند.



- می توان از تعداد بیشتری کرنل های تبدیل استفاده کرد. بعنوان مثال در تصاویر می توان از دو کرنل برای تبدیل لایه های رنگی تصاویر استفاده کرد و هر کدام از آنها یک لایه جدید ایجاد می نمایند.



- در کنار ماتریس ضرایب کرنل یک عدد قابل یادگیری ثابت بایاس (Bias) نیز وجود دارد که می توان بعنوان یکی از پارامترهای سیستم از آن یاد کرد.

- حاصلضرب ماتریس های سه بعدی بالا در تعداد کرنل های مشخص، باعث ایجاد همان تعداد (به تعداد کرنل ها) ماتریس می شود که به آنها کانال (Channel) گفته می شود. در واقع اعداد این ماتریس ها خصوصیاتی هستند که محاسباتی روی آنها انجام شده است و با رنگ های قرمز و سبز و آبی (RGB) متفاوت هستند.
- در واقع سه کانال اصلی و اولیه تصاویر کانال های قرمز و سبز و آبی هستند که با ضرب در ماتریس ضرایب کرنل تبدیل به کانال های جدیدی می شوند.
- با توجه به اینکه عدد حاصلضرب ماتریس کرنل در ماتریس داده ها بصورت یک عدد بدست می آید، معمولا ابعاد ماتریس ضرایب کرنل را بصورت عدد فرد انتخاب می کنند (۳ و ۵ و ۷و ..) که مرکز یکتایی داشته باشد و دقیقا در مرکز ماتریس واقع شود.
- به ماتریس های ضرایب کرنل ماتریس های فیلتر یا فیلتر کانولوشن نیز گفته می شود.
- در واقع برای تصاویر مختلف از کرنل های ثابت استفاده می شود و لذا پس از ضرب آنها در ماتریس داده های تصاویر خصوصیات جدید (Features) در تصاویر دیده می شود.
- در فرآیند یادگیری عمیق ماتریس ضرایب بصورت اتفاقی (رندوم) در نظر گرفته می شوند و سپس در طول فرآیند یادگیری ضرایب تغییر می نمایند.
- پس از انجام فرآیند یادگیری، کرنل ها برای همه تصاویر ثابت می مانند.
- استفاده از مدل هایی که کرنل ها در آنها ایجاد شده اند و اصطلاحا از قبل یادگیری شده اند (Pre Trained) به Transfer Learningمعروف است.
- استفاده از کرنل ها در فرآیند یادگیری عمیق بمنظور استخراج خصوصیات جدید است و نه بمنظور دسته بندی یا تصمیم گیری برای داده های ورودی. در واقع از خصوصیات جدیدی که بدست می آید می توان برای یادگیری مدل استفاده نمود.
- ابعاد تنسور های مدل در پای تورچ مساوی حاصلضرب تعداد کانال ها در تعداد پیکسل های عرض در تعداد پیکسل های ارتفاع تصویر می باشد.
- مثلا اگر تصویر رنگی بعلاوه دو کانال جدید که خطوط عمودی و افقی را مشخص نموده است داشته باشیم، ابعاد تنسور ورودی مدل مساوی 5x1675x3000 = 25125000 خواهد بود.

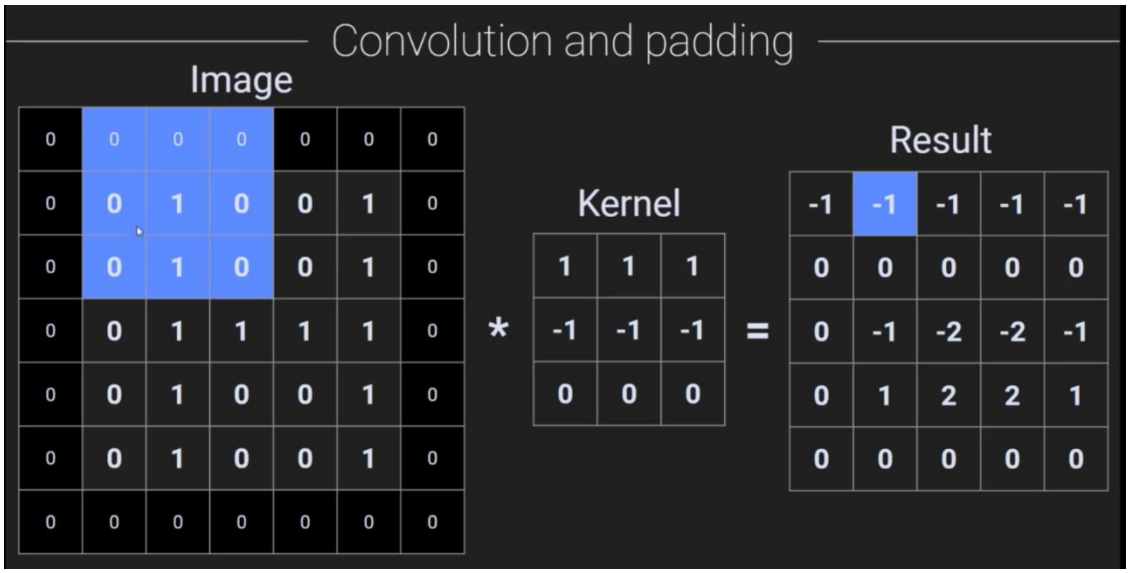
استفاده از سای پای (Scipy convolve2d) و پای تورچ (F.conv2d) در فیلتر کردن تصاویر

- فایل Convolution_1400_12_05.ipynb

- با استفاده از توابع بالا می توان بصورت ساده ماتریس ضرایب کرنل را در تصاویر اعمال (ضرب) کرد.
- سای پای کناره های تصویر (پدینگ) را حذف می کند.

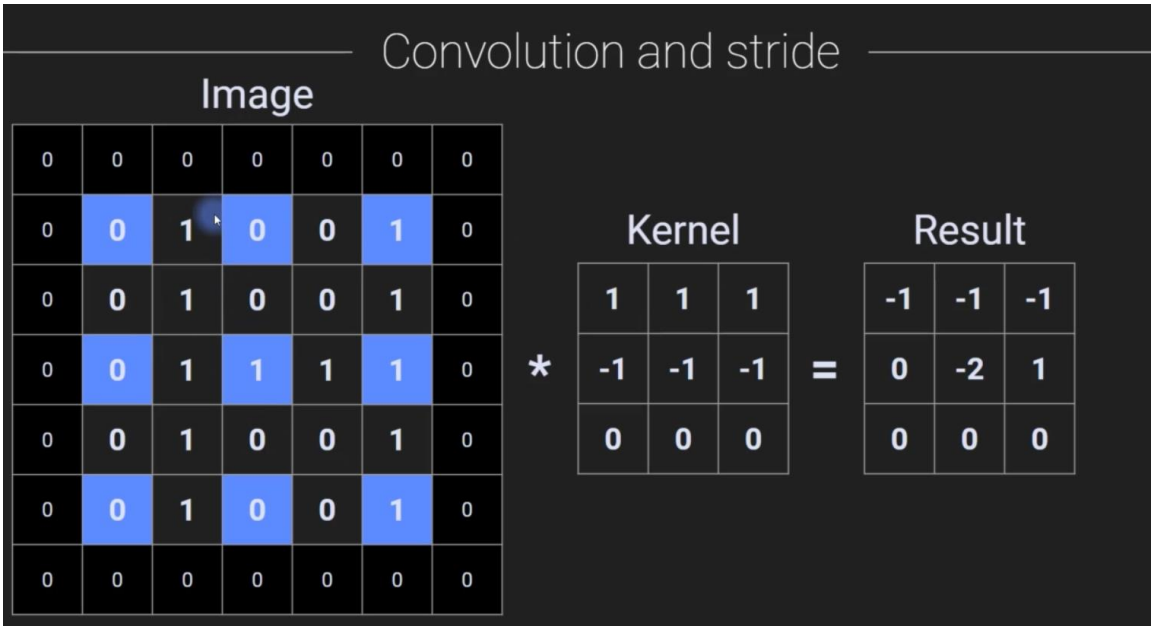
مفهوم پدینگ (Padding)

- استفاده از یک ردیف در هر چهار طرف تصویر باعث می شود تصویر خروجی یعنی حاصلضرب ماتریس کرنل در تصویر اصلی با ابعاد اولیه برابری نماید. تصویر زیر نشان می دهد که یک ردیف در هر وجه با مقدار صفر به ماتریس داده های تصویر اولیه اضافه شده است.

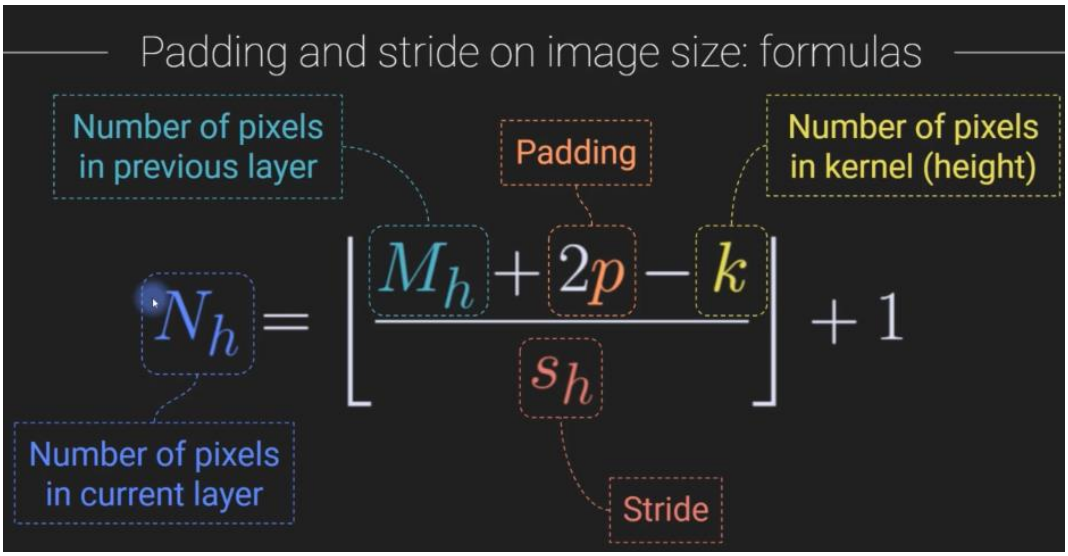


مفهوم استراید (Stride)

- اگر لازم باشد تا ابعاد تصویر کاهش یابد یکی از راهکارها کم کردن تعداد پیکسل های تصویر است. روش استراید کمک می کند تا پیکسل ها را یکی در میان یا چند تا در میان در ماتریس کرنل ضرب کنیم و در نهایت ماتریس حاصلضرب ابعاد کمتری نسبت به تصویر اصلی خواهد داشت.



- در تصویر بالا ابتدا پدینگ و سپس استراید انجام شده است و در نهایت ماتریس خروجی از یک تصویر پنج در پنج و کرنل سه در سه یک ماتریس سه در سه می باشد.
- کاهش ابعاد تصویر باعث کاهش پارامتر های یادگیرنده در شبکه های یادگیری عمیق CNN می شود.
- برای محاسبه اندازه تصویر در یکی از لایه های شبکه عصبی بر اساس میزان پدینگ و استراید از فرمول زیر استفاده می کنیم:



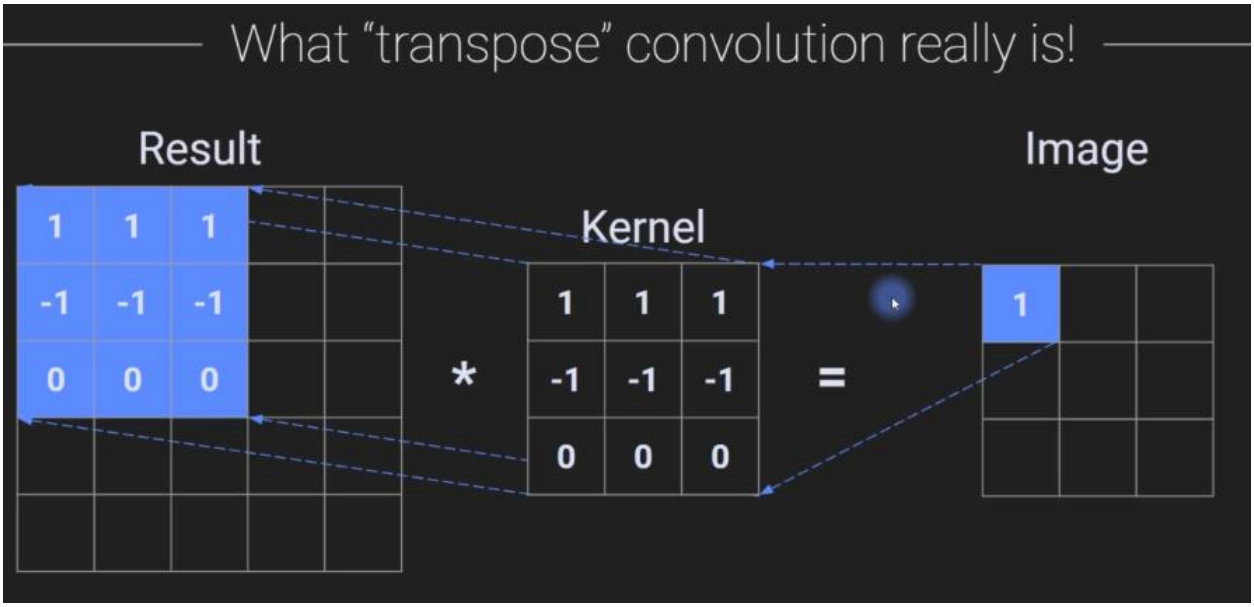
- در این فرمول h به تعداد پیکسل های ارتفاع (ردیف) اشاره دارد.
- وجود براکت های ال شکل (L) در اطراف فرمول به معنی روند کردن به سمت پایین می باشد. یعنی استفاده از جزء صحیح عدد بدست آمده + یک.

استفاده از تابع کانولوشن پای تورچ Conv2

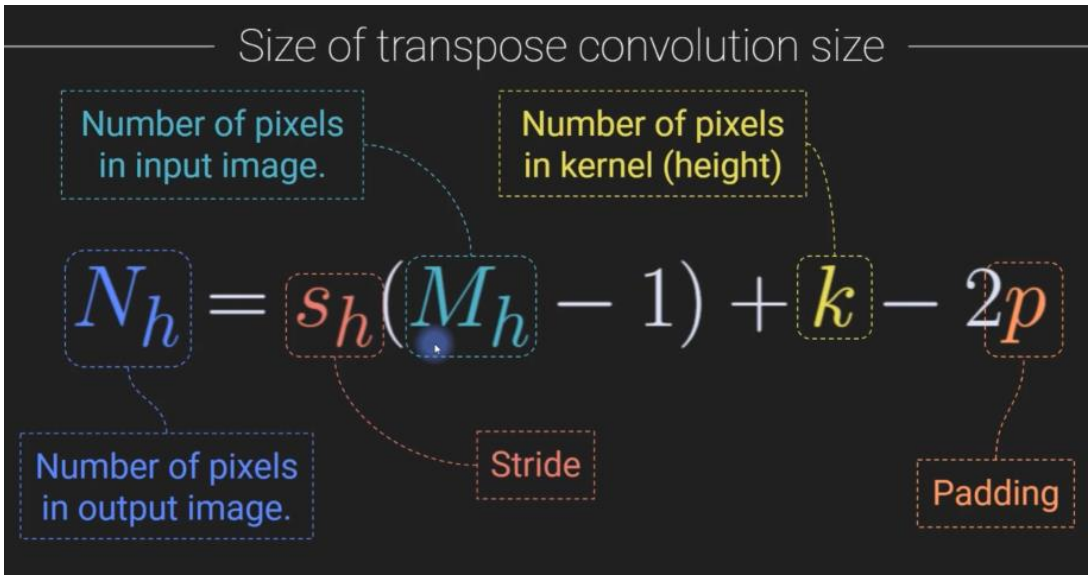
- **فایل Convolution_1400_12_05.ipynb**

مفهوم Transposed Convolution

- این عبارت ممکن است این خطا را در ذهن بوجود آورد که ماتریس ضرایب را بصورت ترنسپوز درآورده و حاصلضرب را محاسبه می کنیم **ولی این درست نیست**.
- در واقع روش محاسبه کانولوشن ترنسپوز این است که برعکس روش پیشروی عمل می کنیم و یک عدد را با استفاده از کرنل به یک ماتریس تبدیل می کنیم. مشابه تصویر زیر.



- این روش بیشتر برای افزایش تعداد پیکسل ها و دقت و رزولوشن در اتوانکودر ها استفاده می شود.
- برای بدست آوردن ابعاد تصویر در لایه ها از فرمول زیر استفاده می نماییم:

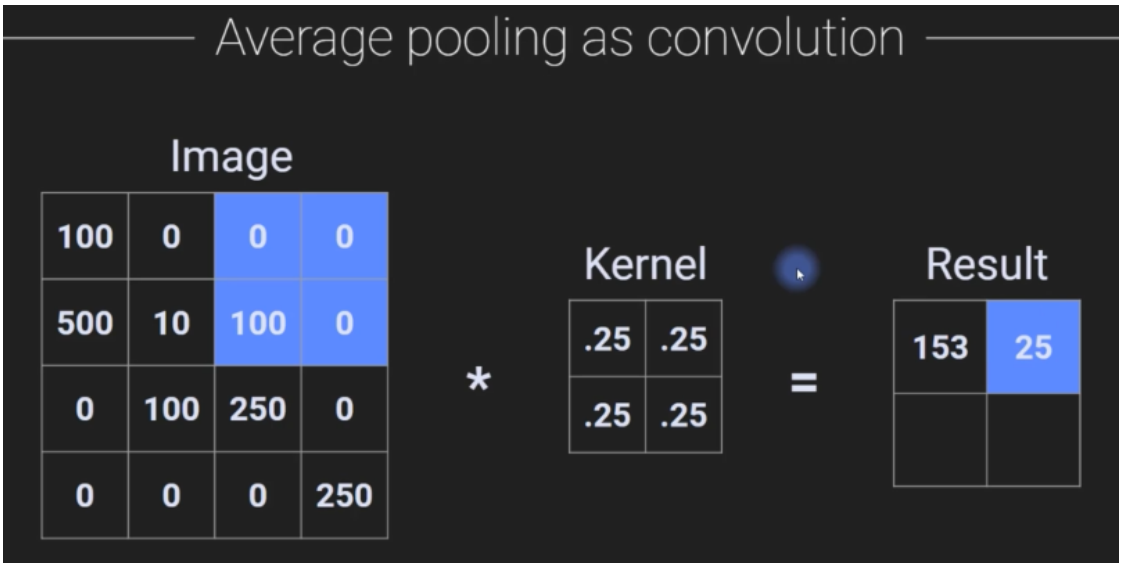


- در پای تورچ از تابع Conv2Transpose2d برای انجام کانولوشن ترنسپوز استفاده می شود

مفهوم Max/mean Pooling

Average Pooling

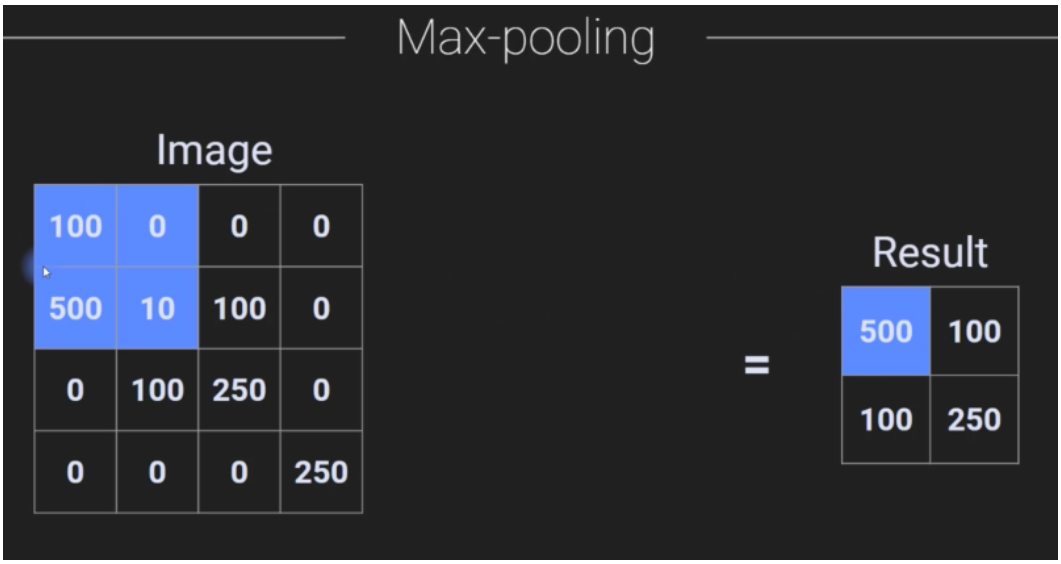
- اگر از یک ماتریس کرنل که محتوای آن میانگین یک عدد باشند مثلا یک ماتریس دو در دو که همه اعضای آن ۰,۲۵ باشند (عدد یک تقسیم بر چهار) برای تبدیل کانولوشن استفاده نماییم، آنگاه در ماتریس حاصلضرب میانگین داده های ماتریس ورودی را برای بخش های دو در دو ماتریس خواهیم داشت.
- نکته اینکه در این فرآیند ماتریس های کوچک تر که از ماتریس اصلی در نظر گرفته می شوند با همدیگر همپوشانی (اورلپ) ندارند و لذا ماتریس خروجی ماتریس کوچک تری خواهد بود. مشابه تصویر زیر.
- به این فرآیند Average(mean) Pooling گفته می شود
- از این روش برای کاهش ابعاد ماتریس و خصوصیات در مدل های یادگیری عمیق استفاده می شود..
- استفاده از اوریج پولینگ در هموار سازی تصاویر استفاده می شود. خصوصا تصاویر نویزی که پیکسل های نامرتبط دارند مثل عکس های آسیب دیده.



Max Pooling

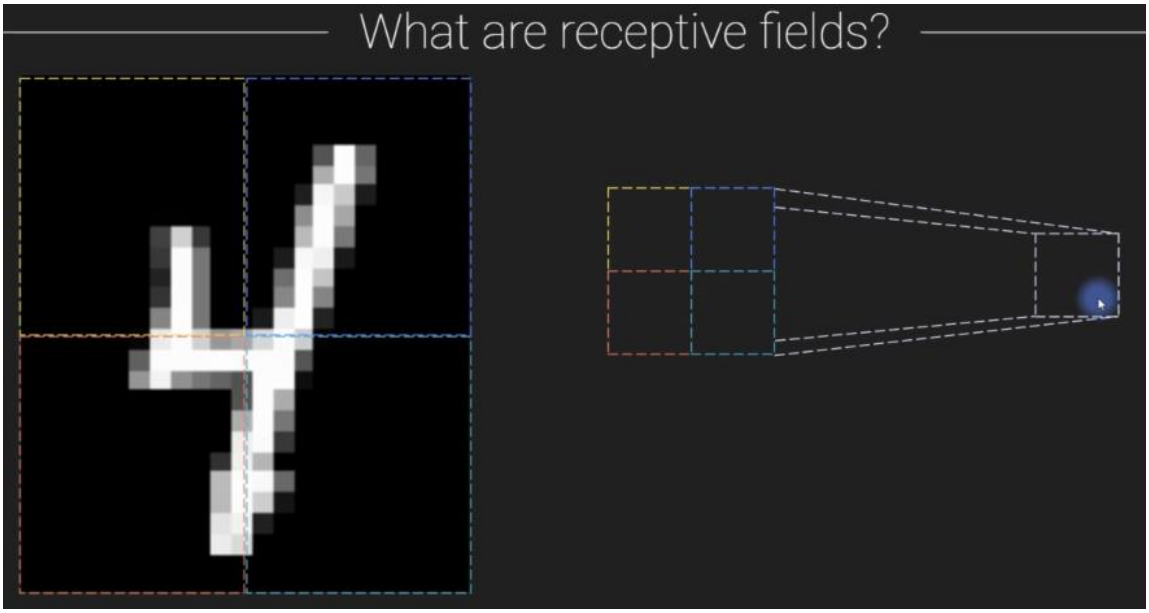
- زمانیکه بخواهیم داده های ماکزیموم را از ماتریس های داده شده کوچک تر (قسمت های دو در دو) که در دل ماتریس داده های ورودی هستند جداسازی کنیم و در یک ماتریس کوچک تر ذخیره کنیم باید از روش مکس پولینگ استفاده نماییم.
- استفاده از مکس پولینگ در مشخص نمودن داده های با تفاوت های زیاد (کانتراست تصویر) استفاده می شود و می توان با این روش کنتراست را زیاد کرد.

- پیدا کردن لبه ها یکی از کاربرد های مکس پولینگ است.
- استفاده از اوریج پولینگ در هموار سازی تصاویر استفاده می شود. خصوصا تصاویر نویزی که پیکسل های نامرتبط دارند مثل عکس های آسیب دیده.

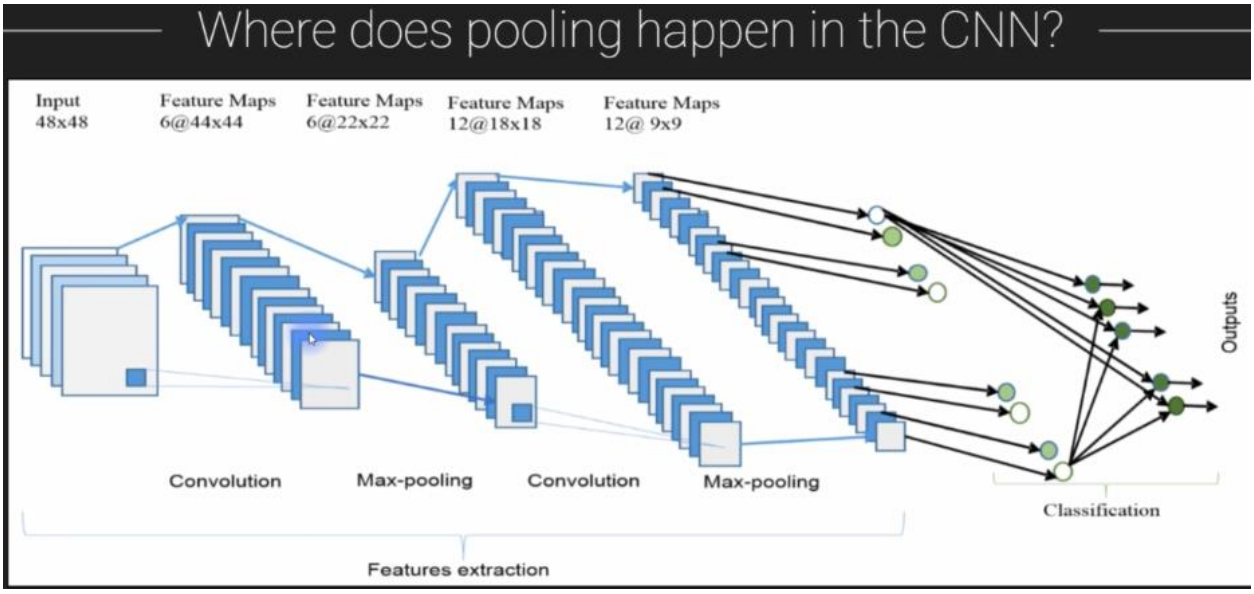


فیلدهای پذیرا (Receptive Fields)

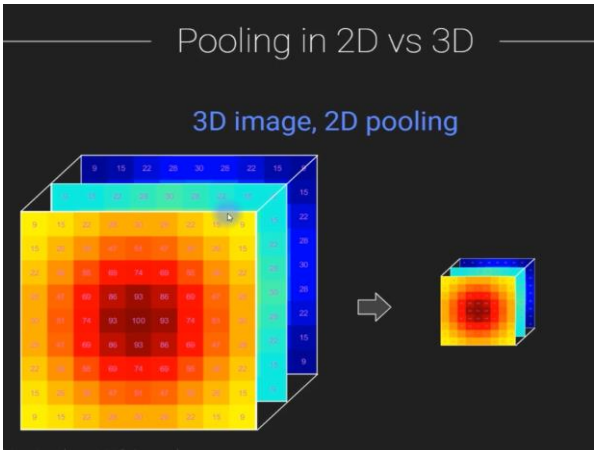
- مفهوم فیلد های پذیرا این است که وقتی از روش های مکس یا مین پولینگ برای کاهش بعد تصاویر استفاده می کنیم، بطور مثال هر چهار سلول ماتریس داده های ورودی به یک سلول تبدیل می شود. یک سلول خروجی (نتیجه مکس پولینگ) بعنوان فیلد پذیرای تصویر ورودی نامیده می شود.

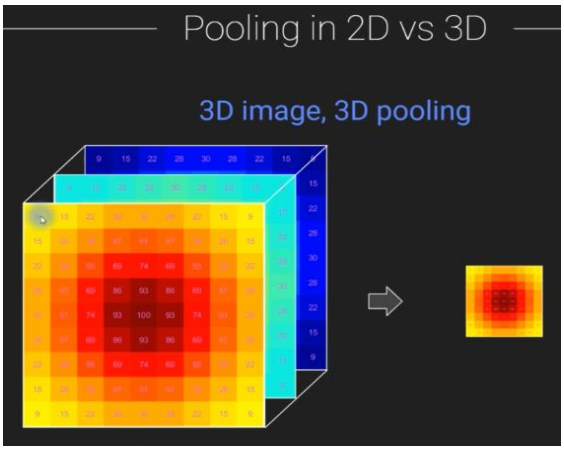


- استفاده از کانولوشن و مکس پولینگ در مدل های سی ان ان بصورت جفتی و در امتداد همدیگر مرسوم است.



- در پولینگ تصاویر با سه لایه رنگی (RGB) می توان از دو روش استفاده نمود.
  - تبدیل تصویر با همان لایه های سه گانه : بدین معنی که هر لایه بصورت جداگانه پولینگ شود و خروجی ماتریس های سه گانه کوچک تر باشند.
  - تبدیل تصویر از لایه های سه گانه به یک لایه : در این حالت ماتریس های بطور مثال دو در دو از لایه قرمز و سبز و آبی که در امتداد هم هستند در نظر گرفته می شود و از ۱۲ سلولی که مشخص شده است فرآیند پولینگ انجام می شود و در نهایت یک عدد خروجی بدست می آید.





مقایسه بین پولینگ و استراید (Pooling vs Stride)

- هر دو روش برای کاهش ابعاد تصویر استفاده می شوند.
- روش کار استراید به این صورت است که از روی تعداد مشخص شده (توسط کاربر) می پرد و در ماتریس ضرایب کرنل ضرب نمی کند.
- روش کار پولینگ این است که تعداد مشخص شده (توسط کاربر) ماتریسی را در ماتریس داده های اصلی جدا و ماکزیموم یا مینی موم یا میانگین آنها را بعنوان یک عدد بر می گرداند.

پولینگ	استراید
از نظر محاسباتی سریعتر است	از نظر محاسباتی کمی کند تر است
پارامتری ندارد	دارای پارامتر های یادگیرنده است
کرنل کوچکتری دارد	کرنل نسبتا بزرگتری دارد
خیلی با ثبات است	ممکن است در مدل های پیچیده دچار اختلال شود

- استفاده از این روش ها در مدل های یادگیرنده عمیق سلیقه ای است و کسی نمی تواند بگوید کدام بهتر است. شاید بهتر باشد که هردو را تست کنیم. روش پولینگ دارای سابقه استفاده بیشتری بوده است.

تبدیل تصاویر (Image Transforms)

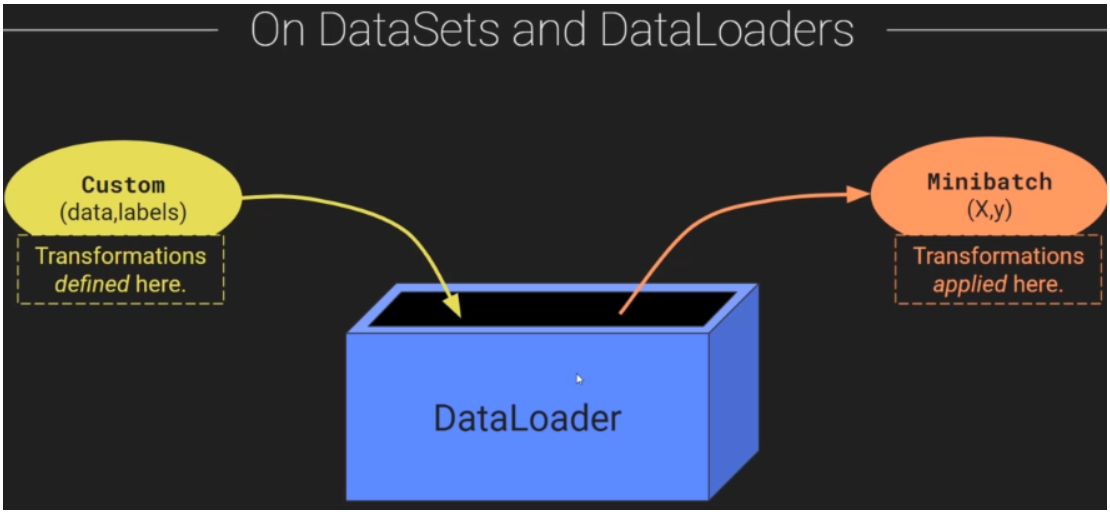
- دو دلیل عمده برای تبدیل تصاویر وجود دارد:
  - مدل های سی ان ان از قبل آموزش دیده (Pre-trained CNNs) با تصاویری با ابعاد مشخص آموزش دیده اند و برای استفاده از آنها باید ابعاد تصاویر تغییر کند ویا بصورت سیاه و سفید به مدل ارائه شود.
  - با تبدیل تصاویر می توان داده های بیشتری برای آموزش مدل بدست آورد. در واقع با تبدیل تصاویر داده های کم ارزش تر کنار گذاشته می شوند و داده های با ارزش تر که محتوای تصاویر را حفظ می کنند باقی می مانند. یکی از روش های تبدیل تصاویر data augmentation است مثل تغییر رنگ، تغییر اندازه، تغییر جهت و ...
  -

تابع ToTensor در torchvision.transforms

- از تابع ToTensor() برای تبدیل تصاویر یا آرایه های نامپای به تنسور استفاده می شود.
- استفاده از این تابع علاوه بر تبدیل ماتریس ها به تنسور، مقیاس داده ها را نیز انجام می دهد و از بازه صفر تا ۲۵۵ به بازه صفر و یک تبدیل می کند.
- علاوه بر آن در آرایه ها ساختار داده ها بصورت (H x W x C) یعنی (کانال رنگ x عرض x ارتفاع) می باشد که در تنسور باید بصورت (C x H x W) تبدیل شود و این تابع آن را انجام می دهد.

استفاده از دیتالودر (DataLoader)

- برای وارد کردن داده ها به مدل سی ان ان لازم است تا داده ها آماده سازی و سپس توسط دیتالودر به مدل وارد شوند.
- ابتدا دیتاست شامل داده ها و لیبل ها باید به تنسور تبدیل شوند و با هم مجتمع گردند.
- سپس داده ها (تصاویر) را تبدیل می کنیم (Transform) ولی تبدیلات را اعمال نمی کنیم.
- پس از آن داده ها را در دیتالودر بارگذاری می کنیم تا بصورت دسته بندی شده (بچ های کوچک) به مدل وارد نماید.
- در زمان بارگذاری در مدل کار اعمال تبدیل تصاویر انجام می شود.



ترتیب انجام کارها برای تصاویر

۱- وارد کردن داده ها
۲- ایجاد یک کلاس برای دیتاست جدید
۳- تعریف تبدیلات تصاویر
۴- ایجاد یک دیتاست برای داده ها و تبدیلات
۵- ایجاد یک دیتالودر

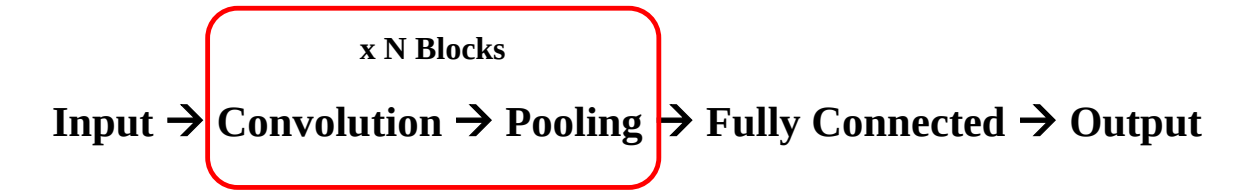
- وقتی از دیتاست CIFAR10 استفاده می کنیم، عملا گام های یک و دو ادغام می شوند.
- زمانیکه می خواهیم داده ای را وارد مدل نماییم باید به تنسور تبدیل شود.
- ترتیب داده ها در آرایه های نامپای (H x W x C) یعنی تعداد کانال x عرض x ارتفاع است و باید به عرض x ارتفاع x تعداد کانال تبدیل شود. (C x H x W)
- تعداد تصاویر نیز برای وارد کردن داده ها به دیتالودر باید مشخص شود که در ابتدای آرایه تنسور وارد می شود.
- لذا در تنسور های ورودی به دیتالودر خواهیم داشت (عرض x ارتفاع x تعداد کانال x تعداد تصاویر)

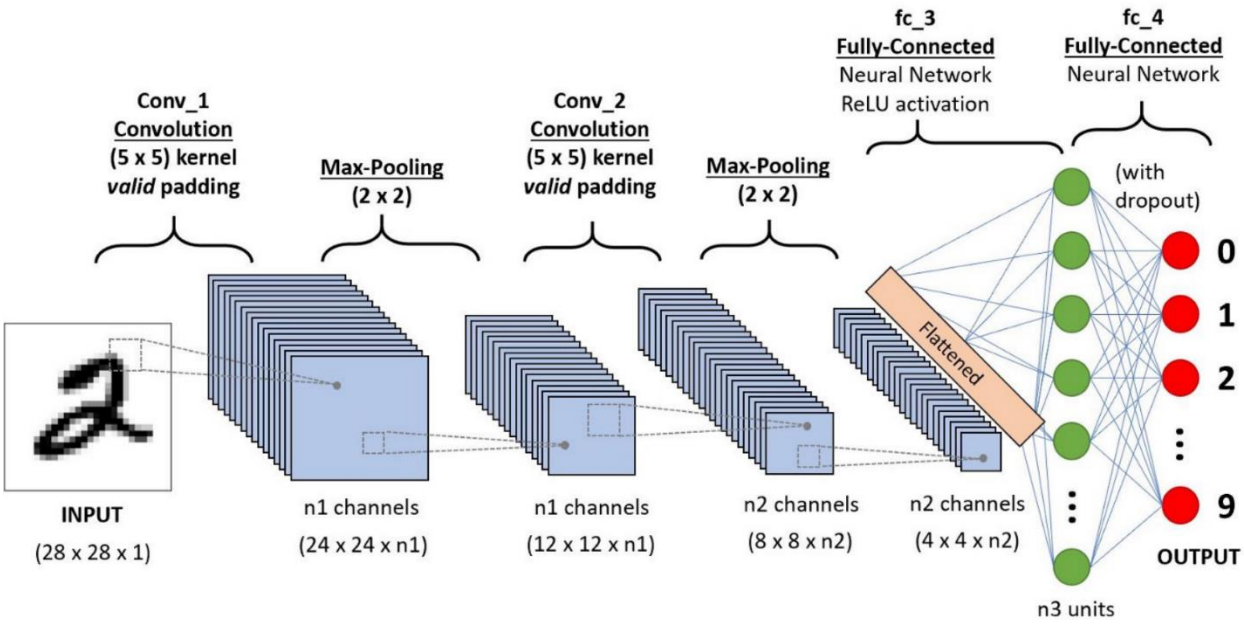
فولدر ۱۹ Understand and design CNNs

شبکه های عصبی کانولوشن (Convolutional Neural Network) عمدتا دارای لایه های زیر هستند:

- **لایه کانولوشن (Convolution)** : یادگیری فیلترها (کرنل یا ماتریس ضرایب) جهت مشخص کردن خصوصیات مهم. در این لایه ماتریس ضرایب بصورت اتفاقی (رندوم) مشخص می شود و سپس در روند برگشت (Back Propagation) ماتریس ها اصلاح و ضرایب داخل آنها یادگیری می شوند.

- **لایه پولینگ (Pooling)** : کاهش ابعاد تصاویر و ماتریس های ورودی و افزایش ابعاد فیلد های پذیرا (Receptive Field)
- **لایه کاملا متصل (Fully connected)** : این لایه کار پیش بینی و دسته بندی را انجام می دهد.





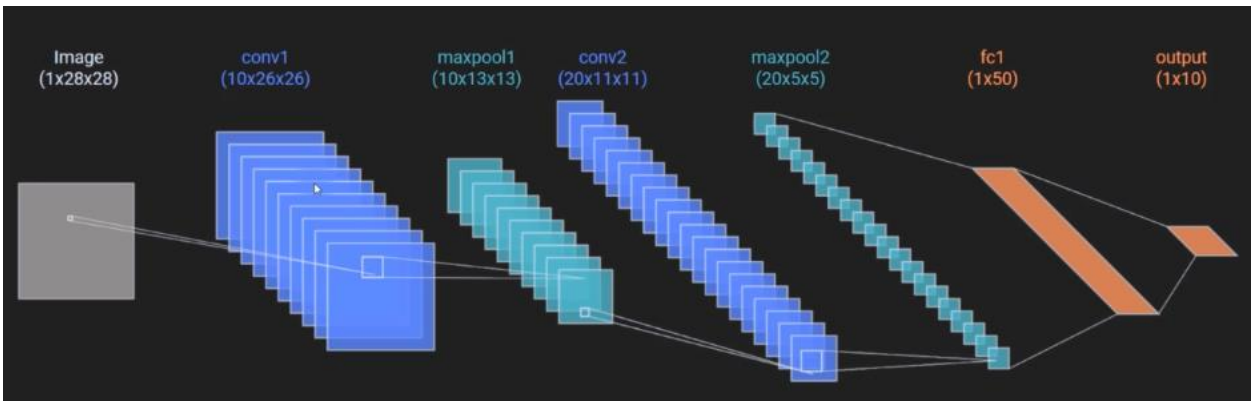
یک منبع خوب برای مطالعه و درک چگونگی عملکرد کرنل و پولینگ:

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

مدل سی ان ان جهت تشخیص اعداد دستنویس (MNIST)

در زیر تصویر کلی مدل ارائه می گردد.

- در این شکل ورودی بصورت یک تصویر سیاه و سفید با ابعاد ۲۸ در ۲۸ ارائه می گردد.
- سپس از ۱۰ ماتریس کرنل برای تبدیل تصویر استفاده می شود که باعث ایجاد ۱۰ تصویر با ابعاد ۲۶ در ۲۶ میشود. کم شدن سایز تصویر به دلیل عملکرد کرنل ها می باشد که بدون پدینگ استفاده می شود.
- سپس از ۱۰ تصویر فوق که خروجی کانولوشن می باشد مکس پولینگ می گیریم که باعث می شود بزرگترین اعداد ماتریس تصاویر انتخاب گردد و ابعاد تصاویر به ۱۳ در ۱۳ پیکسل کاهش یابد.
- سپس از دو کرنل برای تبدیل تصاویر استفاده می کنیم و ۱۰ تصویر بعنوان ورودی به این لایه وارد می شود و با ضرب در دو کرنل تبدیل به ۲۰ تصویر با ابعاد ۱۱ در ۱۱ می گردد.
- مجددا از مکس پولینگ استفاده نموده و تصاویر به ابعاد ۵ در ۵ پیکسل می رسند.
- در لایه بعدی شبکه عصبی کاملا متصل را داریم که ورودی آن ۲۰ تصویر با ابعاد ۵ در ۵ می باشد که باید هموار (Flatten) شده باشد. در واقع ورودی شبکه عصبی (FFN) یک ماتریس با ابعاد ۵۰۰ (20x5x5 = 500) می باشد که به ۵۰ نود متصل می شود.
- در این قسمت پیش بینی انجام می شود و خروجی یکی از اعداد صفر تا ۱۰ خواهد بود.



کدهای این قسمت در فایل زیر قرار دارد:

CNN_02_1400_12_11.ipynb

توضیح مربوط به فرآیند و الگوریتم بخش اول فایل:

CNN to classify MNIST digits

- وارد کردن کتابخانه های لازم
- وارد کردن داده ها از بخش نمونه های داده های گوگل کولب (Sample_data/mnist_train_small.csv)
  - اختصاص ستون اول به لیبل ها و سایر ستون ها به بخش داده ها
  - نرمالایز کردن داده ها
  - تبدیل شکل داده ها به حالت ۴ بعدی زیر

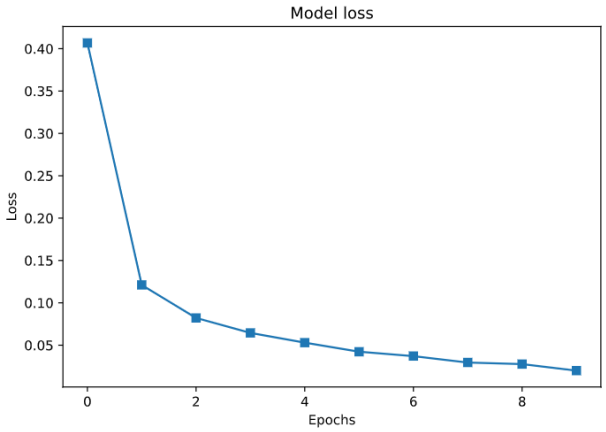
```
(Number of images, number of channels, number of H pixels, Number of W pixels)
```

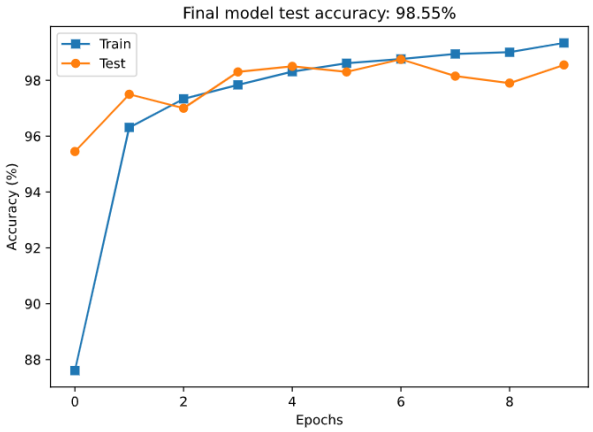
- تبدیل داده ها به تنسور جهت امکان کار با کتابخانه تورچ
  - استفاده از تابع ترین تست اسپلیت از کتابخانه سایکیت لرن
  - یکی کردن داده های آموزش و لیبل ها در یک تنسور با استفاده از تابع TensorDataset
  - ایجاد یک شیء از دیتالودر جهت استفاده در ادامه برنامه با بچ سایز ۳۲ تایی
- نوشتن یک تابع جهت ایجاد شبکه عصبی و قسمت های مختلف آن (یک ورودی جهت پرینت گرفتن یا نگرفتن داریم)
  - ایجاد یک کلاس که از nn.Module ارث می برد و تمام خصوصیات آن را خواهد داشت
  - ایجاد یک مرحله کانولوشن با ۱۰ کرنل که تصویر ورودی سیاه و سفید (۲۸ در ۲۸) گرفته و آن را در ۱۰ کرنل متفاوت ضرب کرده و استراید ۱ و پدینگ ۱ را در نظر می گیرد. ابعاد ماتریس کرنل ۵ در ۵ می باشد.
  - خروجی مرحله اول کانولوشن تصاویر (۱۰ تصویر) با ابعاد ۲۶ در ۲۶ می باشد. اختلاف ابعاد خروجی با ورودی مربوط به فرآیند ضرب ماتریس ها در کناره ها می باشد که با توجه به ابعاد ماتریس ۵ در ۵ یک ردیف از تصویر از کناره ها حذف می شود.
  - ایجاد مرحله دوم کانولوشن که ۱۰ تصویر (خروجی مرحله قبل) را گرفته و به ازای هر تصویر دو کرنل جدید در نظر می گیرد و در نهایت ۲۰ تصویر با همان شرایط کرنل قبلی اجرا می کند.
- **نکته مهم** اینکه در این الگوریتم استفاده از مکس پولینگ در ادامه برنامه و ورود داده ها و آموزش مدل انجام خواهد شد. بدین ترتیب که پس از مرحله اول کانولوشن یک مرحله مکس پولینگ خواهیم داشت که ابعاد تصویر از ۲۶ در ۲۶ به ۱۳در ۱۳ تبدیل می گردد. در مرحله دوم کانولوشن ابعاد تصویر از ۱۳ در ۱۳ به ۱۱ در ۱۱ تبدیل می گردد و در مرحله دوم مکس پولینگ ابعاد تصاویر به ۵ در ۵ تبدیل می گردد.
- در تعریف تابع forward برای هر مرحله کانولوشن یک تابع فعال سازی (activation function) از نوع ReLU استفاده می گردد.
- در نهایت و پس از انجام دو مرحله کانولوشن و مکس پولینگ ۲۰ تصویر با ابعاد ۵ در ۵ پیکسل خواهیم داشت که مجموعا 20x5x5 = 500 پیکسل خواهیم داشت که بعنوان داده های ورودی به شبکه عصبی کاملا متصل (Fully connected feed forward Neural Network یا FFN) وارد می شود.
- در شبکه عصبی تعداد نود ها یا سلول های عصبی (fc1) (در این مساله) ۵۰ عدد در نظر گرفته شده است.
- خروجی این شبکه (out) نیز با ۱۰ عدد سلول یا نود در نظر گرفته شده است که باید مشخص کننده (دسته بندی) عددهای دستنویس باشد.
- تابع خطا (lossfunction) از نوع کراس آنتروپی در نظر گرفته شده است. یادآوری اینکه در مسائلی که بیش از دو گزینه (وجود یا عدم وجود True/False) باید دسته بندی شوند از این نوع تابع خطا استفاده می گردد.

- بهینه ساز این مدل (Optimizer) هم از نوع Adam در نظر گرفته شده است.
- نرخ یادگیری ۰,۰۰۱ لحاظ گردیده است.

- سپس مدل با یک بچ از دیتالودر (۳۲ داده اول) تست می شود.

- در مرحله بعدی یک تابع برای آموزش مدل و ارائه داده ها نوشته شده است.
  - تعداد رفت و برگشت های کلی مدل (epoch) ۱۰ عدد در نظر گرفته شده است.
  - در داخل حلقه epoch حلقه مربوط به آموزش مدل با استفاده از دیتالودر قرار گرفته است.
  - ارزیابی مدل و تشکیل ماتریس داده های خطا در epoch های مختلف نیز (جهت رسم نمودار) انجام می شود.
- در انتها نمودار خطا بر اساس epoch انجام می شود.





استفاده از مدل های سی ان ان در تشخیص اعداد دستنویس که در آنها جابجایی اتفاق افتاده است (Shifted) دارای دقت و عملکرد بالاتری می باشد.

در تعریف مدل ها و توابع باید تا جایی که امکان دارد از پارامتر ها و متغیر ها در برنامه استفاده شود. به این روش Softcoding گویند و بر عکس آن زمانیکه از اعداد بصورت خالص در برنامه استفاده شود که در صورت نیاز به تغییر باید تک تک آنها را تغییر داد Hardcoding گویند.

برای استفاده از GPU باید در کدهای برنامه مشخص کرد که از آن استفاده نماید. موارد زیر باید توجه گردد

- تغییر پردازنده برنامه از CPU به GPU در گوگل کولب یا بستر برنامه نویسی دیگر
- تعریف GPU توسط پای تورچ :

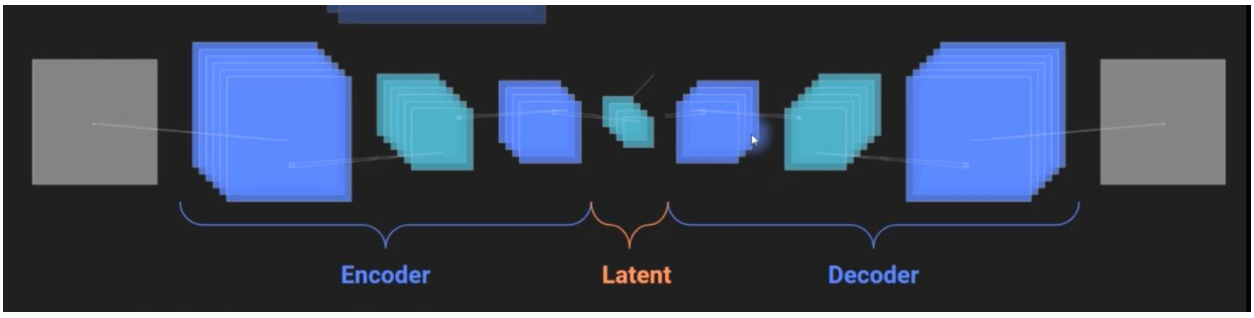
```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

- ارجاع دادن شبکه عصبی و ورودی های داده و لیبل ها (آموزش و تست جداگانه) به GPU

```
net.to(device)
X = X.to(device)
y = y.to(device)
```

استفاده از اتو انکودر در تشخیص مشکلات تصویر

- ساختار اتو انکودر به شکل زیر است:



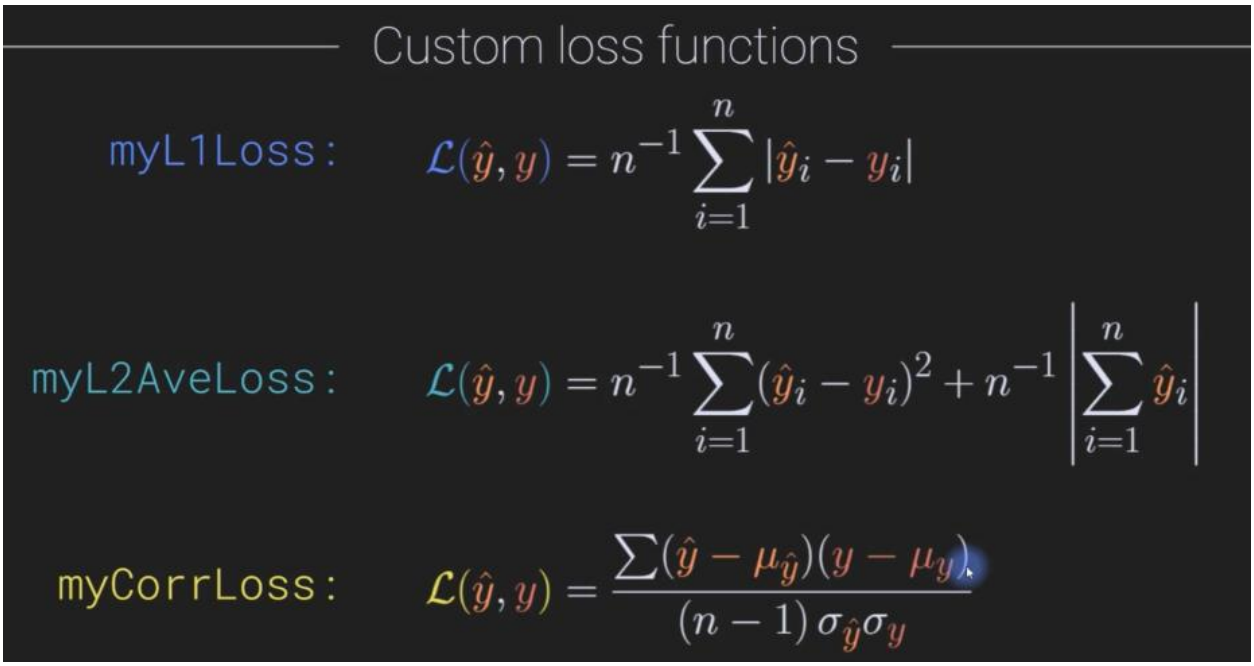
- از یک تصویر بزرگ به تصاویر کوچک تر و عمیق تر حرکت می کنیم و سپس به تصویر بزرگ برگشت می نماییم.
- در قسمت اول (انکودر) و قبل از گلوگاه یا `latent` از کانولوشن استفاده می کنیم و پس از آن (دی کودر)از ترنسپوز کانولوشن استفاده می شود. هدف از استفاده از ترنسپوز کانولوشن افزایش ابعادی تصویر می باشد.
- ضرایب ماتریس ها در دی کودر بسیار شبیه ضرایب در انکودر می باشد.
- در ساختار انکودر و دی کودر از ساختار شبکه های عصبی استفاده می شود و ساختار آنها شبیه سی ان ان می باشد.
- در انکودر از کانولوشن و تابع فعال ساز و همچنین مکس پولینگ استفاده می شود.
- در دی کدر از ترنسپوز کانولوشن (`ConvTranspose2d`) و تابع فعال سازی استفاده می شود.

```python
# encoding layer
self.enc = nn.Sequential(
    nn.Conv2d(1,6,3,padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2,2),
    nn.Conv2d(6,4,3,padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2,2)
    )

# decoding layer
self.dec = nn.Sequential(
    nn.ConvTranspose2d(4,6,3,2),
    nn.ReLU(),
    nn.ConvTranspose2d(6,1,3,2),
    )
```

کد ها در فایل **CNN_02_1400_12_11.ipynb**

ایجاد تابع خطا دلخواه (Loss function)



```python
# L1 loss function

class myL1Loss(nn.Module):
  def __init__(self):
    super().__init__()

  def forward(self,yHat,y):
    l = torch.mean( torch.abs(yHat-y) )
    return l
```

```python
# L2+average loss function

class myL2AveLoss(nn.Module):
  def __init__(self):
    super().__init__()

  def forward(self,yHat,y):
    # MSE part
    l = torch.mean( (yHat-y)**2 )

    # average part
    a = torch.abs(torch.mean(yHat))

    # sum together
    return l + a
```

```python
# correlation loss function

class myCorLoss(nn.Module):
  def __init__(self):
    super().__init__()

  def forward(self,yHat,y):

    meanx = torch.mean(yHat)
    meany = torch.mean(y)

    num = torch.sum( (yHat-meanx)*(y-meany) )
    den = (torch.numel(y)-1) * torch.std(yHat) * torch.std(y)
    return -num/den
```

- علامت منفی در تابع خطای آخر (ضریب همبستگی) به دلیل این است که هرچه میزان همبستگی بیشتر باشد بهتر است ولی در اینجا چون تابع خطا نوشته ایم، هرچه میزان خطا کمتر باشد مطلوب است.
- در کل ضریب همبستگی تابع خطای خوبی نیست. در اینجا برای مثال حل شده است.
-

تشخیص حروف الفبای دستنویس (EMNIST)

توضیحات مربوط به کدهای تشخیص حروف الفبا از تصاویر دستنویس .

کدها در فایل CNN_02_1400_12_11.ipynb قرار دارند.

- ابتدا کتابخانه های مورد نیاز وارد می شوند
- سپس GPU را برای برنامه تعریف می کنیم
- داده های مربوط به حروف دستنویس از کتابخانه پی تورچ قابل دانلود هستند. EMNIST
- پس از دانلود، محتوای داده ها را بررسی می کنیم.
  - مشاهده می شود که در داده ها کلاسی تحت عنوان N/A وجود دارد که این مورد در ادامه کار و زمانیکه داده ها باید بصورت وان هات تقسیم بندی شوند ایجاد مشکل خواهد کرد.
  - شکل داده ها (Shape) نیز بصورت سه تایی شامل تعداد تصاویر و تعداد پیکسل های ارتفاع و پیکسل های عرض می باشد که باید بصورت چهارتایی یعنی تعداد تصاویر، تعداد کانال ها و تعداد پیکسل های ارتفاع و پیکسل های عرض تبدیل شود که با استفاده از دستورات پای تورچ view این کار انجام می شود.
- همانطور که در بالا گفته شد باید کلاس N/A حذف شود. برای انجام آن مطمئن می شویم که داده ای در این کلاس موجود نباشد . سپس آن را حذف می کنیم. شماره ایندکس برای داده های موجود باید از صفر شروع و به ۲۵ ختم شود چراکه ۲۶ حرف انگلیسی داریم.
- سپس داده ها را در حالت عادی و نرمال شده ترسیم می کنیم.
- برای دیدن تصاویر آن ها را ترسیم می کنیم.
- از توابع سایکیت لرن برای تقسیم بندی داده ها استفاده می کنیم و سپس داده ها و لیبل ها (آموزش و تست) را یکپارچه می کنیم (با تابع TensorDataset) و سپس دیتالودر را ایجاد می کنیم تا داده های آموزش را بصورت بچ به بچ به مدل وارد کند و پس از آموزش داده های تست را بصورت یکباره به مدل وارد نماید.
- ایجاد مدل یادگیری عمیق
  - در اینجا مثل توابع قبلی از nn.Module ارث بری می کنیم و توابع کانولوشن و نیز توابع نرمال سازی را ایجاد میکنیم.
  - در استفاده از تابع نرمال سازی nn.BatcchNorm2d(k) باید تعداد کانال ها یا فیچر هایی که در لایه قبلی (کانولوشن و مکس پول) ایجاد شده است را بعنوان ورودی (k) بدهیم.
  - سپس روی داده هایی که به ترتیب کانولوشن، مکس پول و نرمال شده اند یک تابع فعال ساز ReLU از نوع leaky_relu اعمال می کنیم.
  - موارد بالا (کانولوشن، مکس پول و نرمال سازی و یک تابع فعال ساز) را یک بار دیگر با توجه به شکل ماتریس های بدست آمده اجرا می کنیم.
  - پس از مشخص کردن توابع قسمت های سی ان ان مدل، قسمت پیش بینی کننده (FFN) را ایجاد می کنیم. در اینجا دو لایه خطی nn.Linear با تعداد ورودی ۲۹۴ در لایه اول و خروجی ۵۰ و لایه دوم با ورودی ۵۰ و خروجی ۲۶ (به تعداد حروف الفبا) ایجاد می کنیم.
  - برای اینکه ورودی این دولایه از حالت دو بعدی به یک بعدی تبدیل شود، مجددا از تابع view استفاده می کنیم.
  - در لایه اول شبکه عصبی FFN از یک تابع فعال ساز leaky_relu استفاده کرده و سپس داده ها را به لایه دوم انتقال می دهیم.

- برای تابع خطا از کراس آنتروپی استفاده می کنیم چراکه چندین انتخاب باید در خروجی مشخص شود.
- بهینه ساز نیز Adam و با نرخ یادگیری ۰,۰۰۱ خواهد بود.

- تابع آموزش مدل
  - برای آموزش از ۱۰ دوره epoch استفاده می کنیم.
  - شبکه تشکیل شده و نیز داده های ورودی و لیبل ها باید به GPU معرفی شوند. با استفاده از دستور .to(device) این کار را انجام می دهیم.
  - در این مثال بجای accuracy از معیار error rate استفاده شده است. در واقع بجای سنجش میزان تشخیص های درست، تشخیص های نادرست در نظر گرفته می شوند
  - یک نکته مهم در بکار گیری داده ها پس از آموزش مدل و استفاده از آنها مثلا در ترسیم ماتری خطا Confusion Matrix این است که داده ها باید به CPU ارجاع داده شوند در غیر این صورت با خطا مواجه می شویم.

```
1 import sklearn.metrics as skm
2
3 # compute the confusion matrix
4 C = skm.confusion_matrix(y.cpu(),torch.argmax(yHat.cpu(),axis=1),normalize='true')
```

مفهوم دراپ اوت (Drop Out in Convolution)

زمانیکه تعدادی از نودهای شبکه عصبی (معمولا FFN) را کاهش می دهیم و اصطلاحا از مدار خارج می کنیم دراپ اوت انجام داده ایم.

- دراپ اوت در واقع برای متسقل بودن نودها و عدم ایجاد اورفیت مهم و کارآمد است.
  - کاهش تعداد پارامترهای شبکه عصبی که نیاز به محاسبات دارند نیز دلیل دیگری در دراپ اوت است.
  - معمولا از ضریب ۵۰ درصد یا همین حدود برای دراپ اوت استفاده می شود. این بدین معنی است که احتمال حذف (خاموش شدن) یک نود از فرآیند محاسبات ۵۰ درصد باشد.

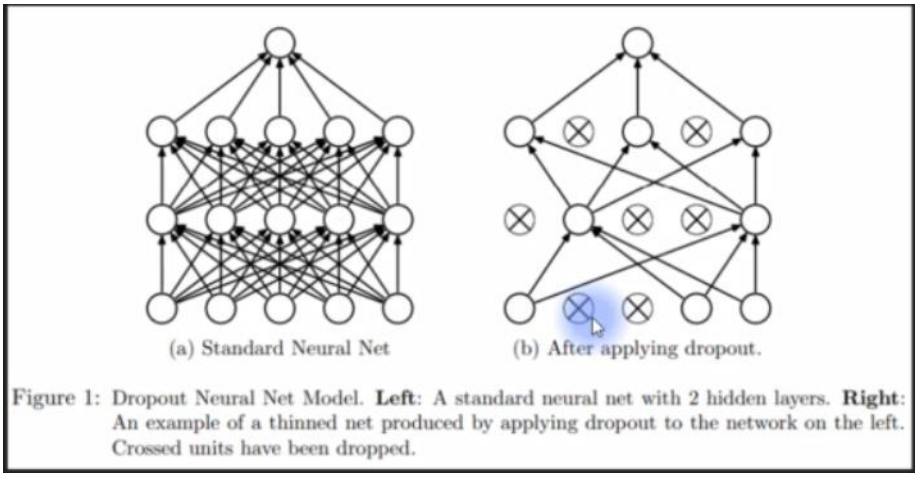


Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- نکته مهم اینکه در شبکه های سی ان ان تعداد پارامتر ها و همچنین احتمال اورفیت بسیار کم است.
- **نکته مهم دیگر** اینکه در لایه های سی ان ان استقلال وجود ندارد و با توجه به اینکه ماتریس ضرایب کرنل بین لایه ها عمل می کنند، در واقع هر لایه به لایه قبلی خود وابسته است و استفاده از دراپ اوت (۵۰ درصد) مشکلاتی ایجاد می کند.
- استفاده از دراپ اوت حدود ۱۰ الی ۲۵ درصد در شبکه های سی ان ان باعث ایجاد نویز روی تصاویر می شود که احتمال اورفیت را به شدت کاهش می دهد.

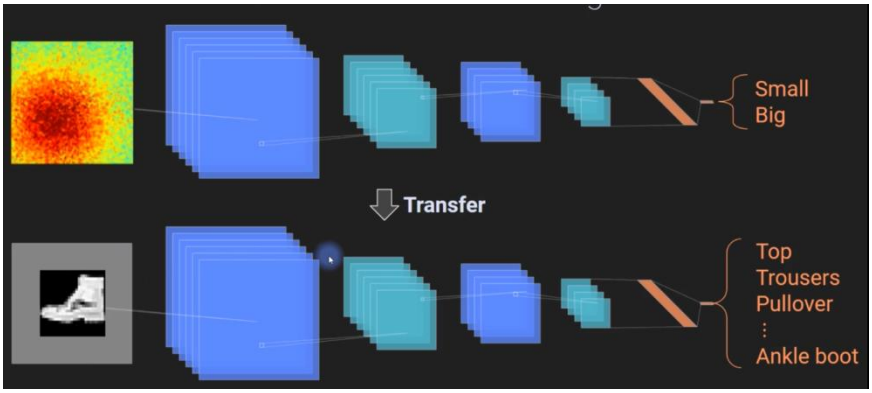
فولدر ۲۱ : Transfer Learning

نام فایل

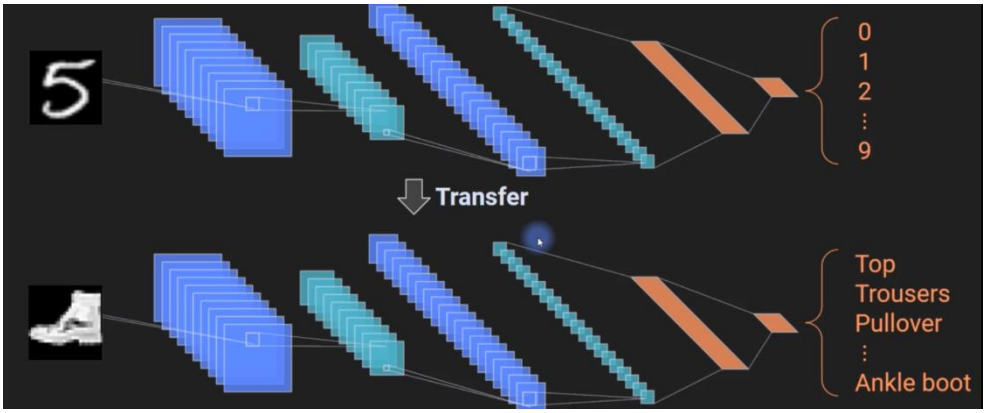
Transfer_Learning_1400_12_21.ipynb

- مفهوم ترنسفر لرنینیگ این است که از مدل شبکه عصبی که توسط کاربر دیگر (یا خودمان) توسط داده های زیاد آموزش داده شده و وزن های آن ها نیز مشخص شده است استفاده کنیم و برای حل مساله جدید بکار بگیریم.
- البته در این روش باید اصلاحاتی (Tuning) در زمینه خروجی (تعداد و نوع آن) صورت پذیرد.
- با توجه به اینکه مدل برای داده های دیگر طراحی و تست شده است، احتمال اینکه برای مساله جدید کارآمد نباشد زیاد است.
- **نکته مهم** اینکه نوع ورودی (ابعاد تصویر) و کاربری مدلی که از قبل تهیه شده است باید بررسی و متناسب با آن برای حل مساله جدید در نظر گرفته شود. مثلا اگر مدلی با ابعاد تصویر ۳۲ در ۳۲ و برای دسته بندی باینری (دو خروجی) تعریف شده است و کارآمدی خوبی دارد، این مدل برای تصاویر ورودی ۲۸ در ۲۸ و دسته بندی ۱۰ تایی مناسب نیست. در تصاویر زیر مثال نامناسب و مناسب دیده می شود.

مدل نامناسب برای مساله جدید (Gaussian circles size و FashionMNIST)



مدل مناسب برای مساله جدید (MNIST و FashionMNIST)



- یک مثال خوب برای استفاده از ترنسفر لرنینگ این است که از مدل AlexNet برای تشخیص مدل بی ام دبلیو استفاده کنیم.
- مثال دیگر استفاده از مدل آموزش دیده تشخیص اعداد (MNIST) و استفاده از آن برای تشخیص نوع لباس (FashinMNIST) می باشد.
- زمانیکه می خواهیم یک مدل آموزش دیده را برای مساله جدید در نظر بگیریم باید پارامترهای آن را روی مدل مساله جدید کپی کنیم. برای این کار باید از دستور زیر استفاده کنیم :

```
for target, source in zip(NewModel.named_parameters(), SourceModel.named_parameters()):
    target[1].data = copy.deepcopy( source[1].data )
```

- در واقع اگر از دستور deepcopy بصورت تنها استفاده کنیم بسیاری از پارامترها به درستی کپی نمی شوند و باید پارامترها با دستورات فوق از مدل آموزش دیده قبلی (SourceModel) به مدل جدید NewModel منتقل شوند.

- در بعضی موارد تعداد خروجی های مدل ها (تعداد دسته بندی مثلا تعداد دیجیت ها و تعداد حروف الفبا) با هم متفاوت است و باید لایه آخر شبکه عصبی تغییر کند. برای انجام این کار پس از کپی کردن پارامتر های مدل مبدا (Source) با دستورات بالا، باید لایه آخر را تغییر دهیم.
- برای این کار ابتدا مشخصات مدل را با دستور پرینت (نام مدل) می گیریم. print (NewModel_name)
- در اطلاعات پرینت شده آخرین لایه را بررسی و تعداد خروجی های آن را بررسی می کنیم. مطابق تصویر زیر:

```
[15]  1 # check out the network
      2 print(numberNet)
      3 print(' ')
      4
      5 # and the final layer
      6 print(numberNet.fc2)
      7
      8 # replace the final layer to have 10 outputs instead of 26
      9 numberNet.fc2 = nn.Linear(50,10)
     10
     11 # and check it again
     12 print(' ')
     13 print(numberNet)
```

```
emnistnet(
  (conv1): Conv2d(1, 6, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bnorm1): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(6, 6, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bnorm2): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=294, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=26, bias=True)
)

Linear(in_features=50, out_features=26, bias=True)

emnistnet(
  (conv1): Conv2d(1, 6, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bnorm1): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(6, 6, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (bnorm2): BatchNorm2d(6, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=294, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)
```

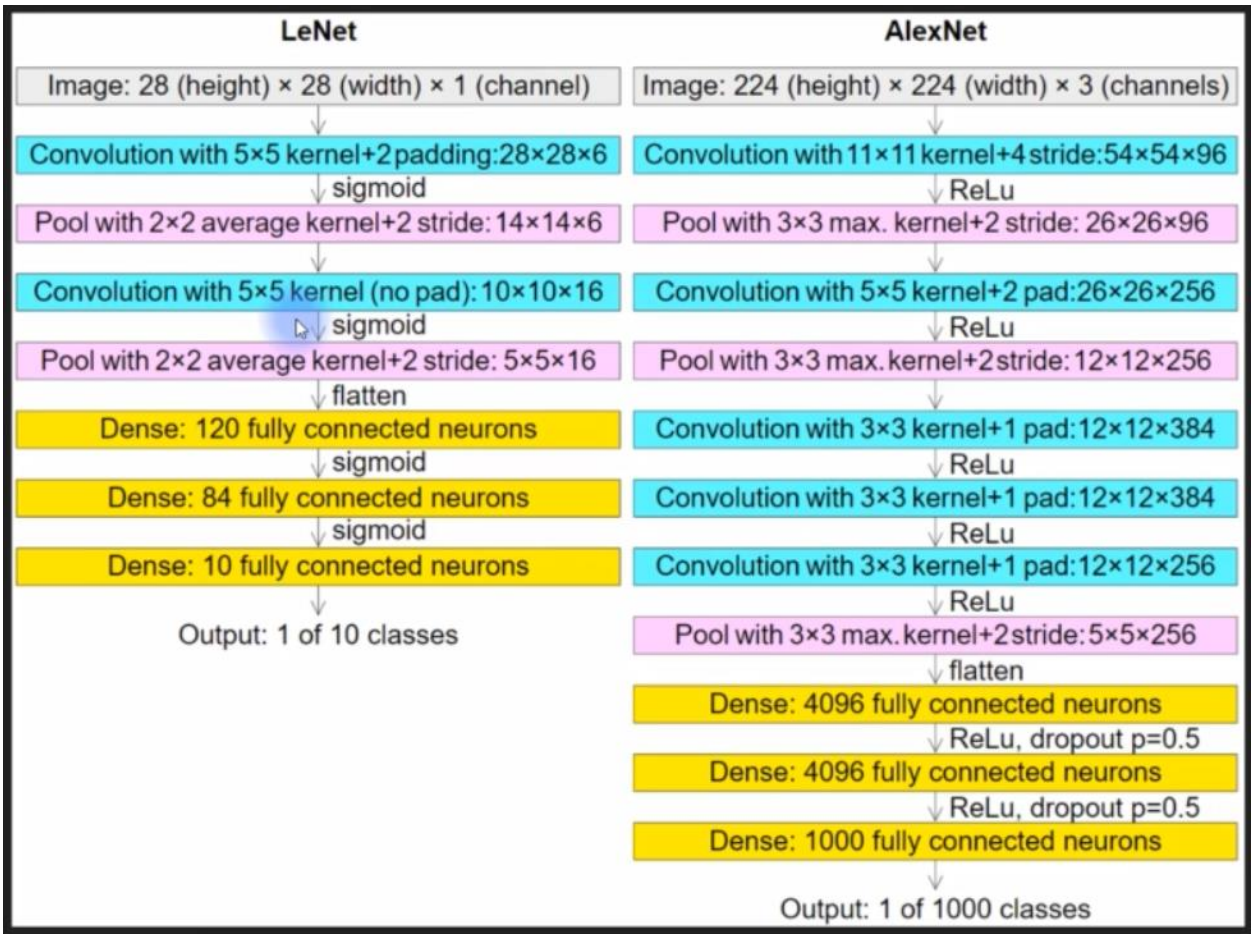
- سپس با فراخوان کردن آن لایه در مدل جدید، تعداد خروجی مورد نظر خودمان را وارد می کنیم. با این کار فقط لایه آخر مدل تغییرات خواهد داشت.
- پس از آن می توانیم مدل جدید را بصورت کامل آموزش داده و مقدار خطا و دقت آن را بررسی کنیم.

روش دیگر استفاده از مدل های آموزش دیده این است که در صورت نیاز به تغییرات در لایه آخر، پس از کپی کردن کلیه پارامترها با روش های فوق الذکر و انجام تغییرات در تعداد خروجی، لایه های میانی را فریز (freeze) کرده و فقط لایه آخر را آموزش دهیم. با این کار نیازی به آموزش مدل جدید بصورت کامل نمی باشد.

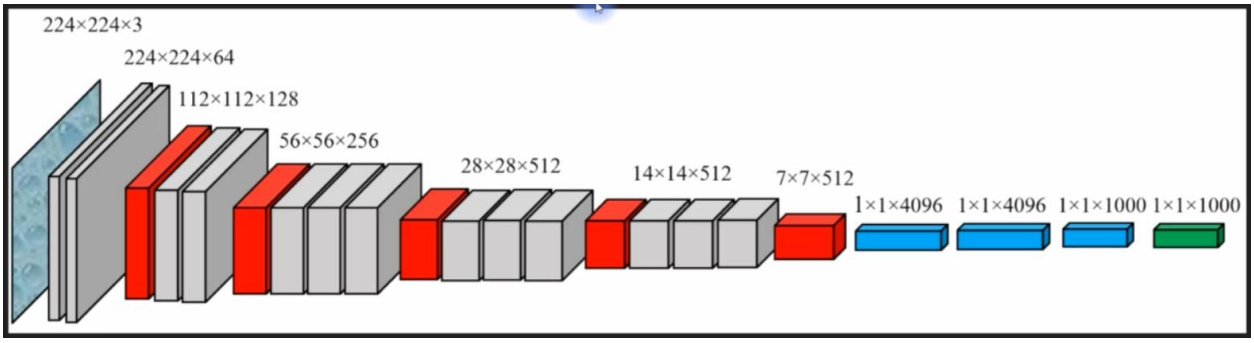
- برای انجام تغییرات و سپس فریز لایه های میانی باید از حلقه ای استفاده کنیم که بجز لایه آخر، یادگیری را متوقف کند.
- لذا از مشخصه requires_grad استفاده می کنیم.

```
# freeze all layers except output
for p in NewModel.named_parameters():
      if not 'fc2' in p[0]:   # fc2 is the output layer in this example
            p[1].requires_grad = False
```

مدل های پر کاربرد CNN : AlexNet و LeNet



VGG Net



VGG Net

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

مدل های پیش آموزش دیده در پای تورچ