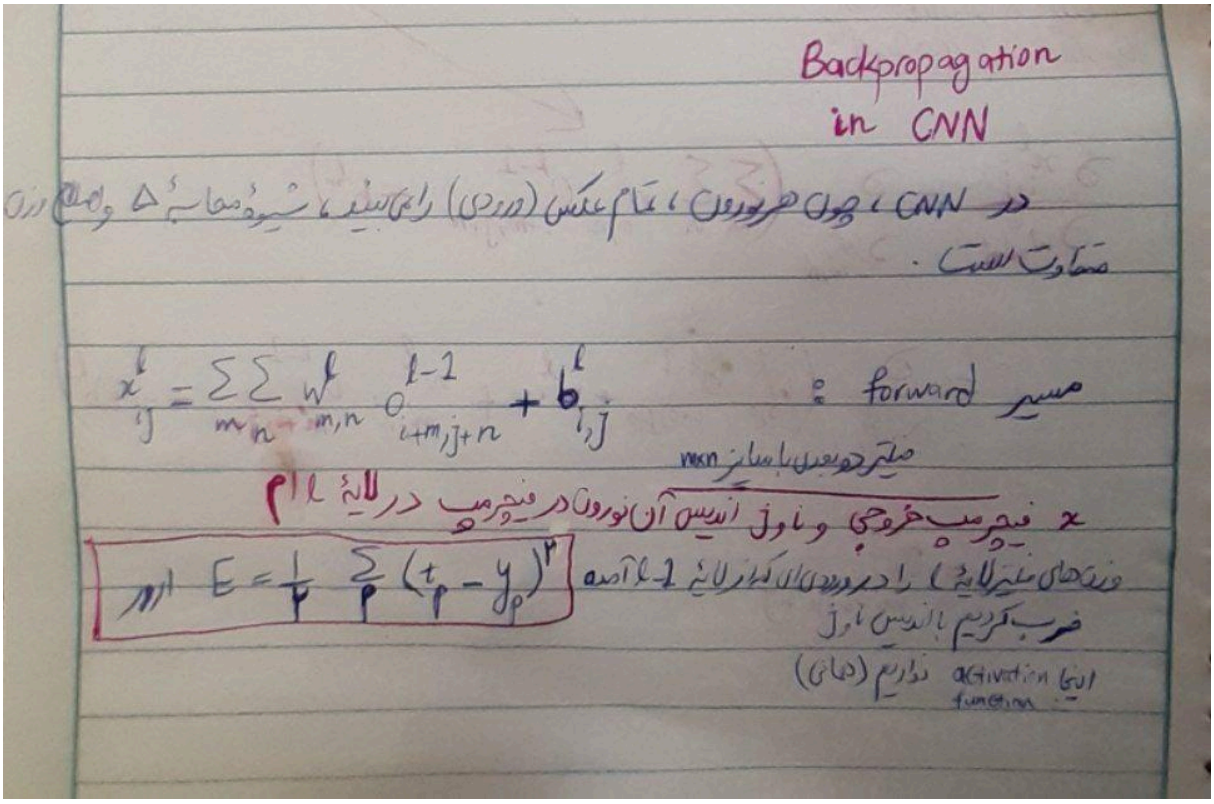
سارا حسینی ۴۰۰۲۲۲۰۲۶

سوال۱-

Gradient accumulation روشی مناسب دیتاهای بزرگ است که استفاده از حافظه را بهینه‌تر میکند. در واقع بجای اینکه بعد از عبور هر بچ دیتا، پارامترها اپدیت شوند، چند بچ عبور میکنند و گرادیان روی هم جمع میشود و در آخر آپدیت میشود. به این صورت، مموری موردنیاز برای ذخیره گرادیان ها کاهش می یابد. مثلا میتوانیم ۴ بچ را به مدل بدهیم و سپس مدل را اپدیت کنیم. وقتی دیتای ما بسیار بزرگ است، مقدار بچ توسط مقدار حافظه‌ای که داریم، محدود میشود. و همین باعث میشود سایز بچ را مجبور شویم کوچک انتخاب کنیم و مدل دیرتر همگرا میشود. با استفاده از تکنیک Gradient accumulation، هر بچ مدنظرمان را به چند مینی بچ تقسیم میکنیم که روی مموری جا بشوند، اما اپدیت را بعد از اینکه همه گرادیان‌های مینی بچ‌ها محاسبه شد، انجام میدهیم. در واقع effective batch size یعنی سایز بچ واقعی را افزایش میدهیم (با تقسیم به مینی بچ، یک بچ بزرگتر را شبیه سازی میکنیم)

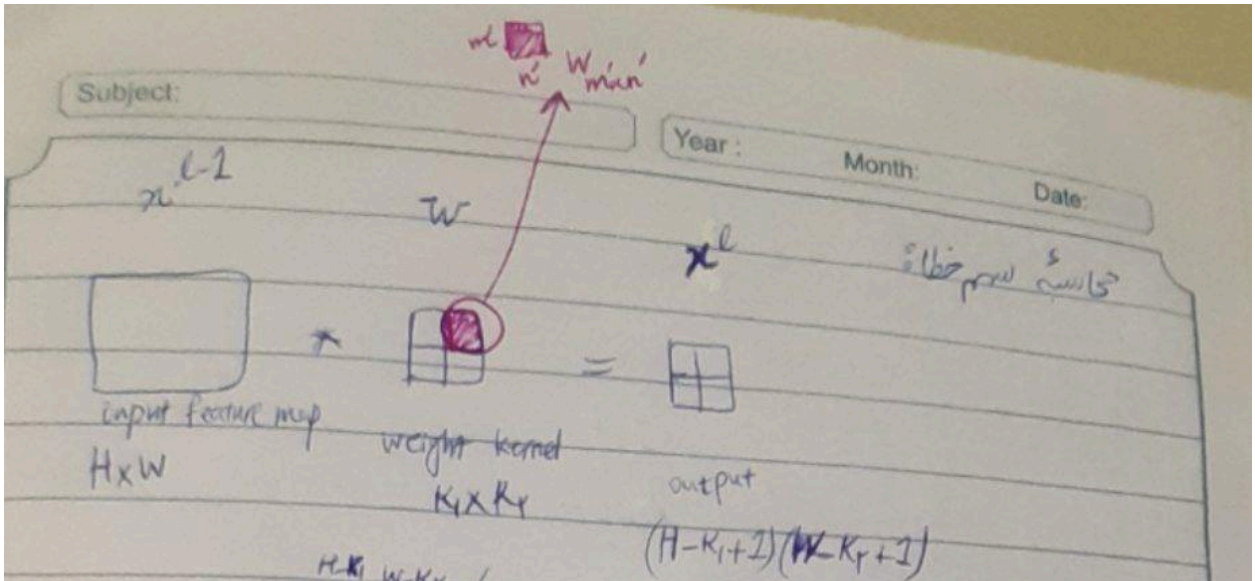
```python
# from https://kozodoi.me/blog/20210219/gradient-accumulation

# batch accumulation parameter
accum_iter = 4

# loop through enumaretad batches
for batch_idx, (inputs, labels) in enumerate(data_loader):
    # extract inputs and labels
    inputs = inputs.to(device)
    labels = labels.to(device)

    # passes and weights update
    with torch.set_grad_enabled(True):
        # forward pass
        preds = model(inputs)
        loss = criterion(preds, labels)

        # normalize loss to account for batch accumulation
        loss = loss / accum_iter

        # backward pass
        loss.backward()

        # weights update
        if ((batch_idx + 1) % accum_iter == 0) or (batch_idx + 1 == len(data_loader)):
            optimizer.step()
            optimizer.zero_grad()
```

سوال۲-


Backpropagation
in CNN
در CNN ، چون هر نورون ، تمام عکس (ورودی) را می‌بیند ، شیوه‌ی محاسبه‌ی Δ و اصلاح وزن
متفاوت است .
مسیر forward :
$x_{ij}^{l} = \sum_{m}\sum_{n} w_{m,n}^{l}\, o_{i+m,j+n}^{l-1} + b_{i,j}^{l}$
فیلتر دو بعدی با سایز mxn
x فیچرمپ خروجی و ناول اندیس آن نورون در فیچرمپ در لایه‌ی l ام
$E = \frac{1}{2}\sum_{p}(t_p - y_p)^2$
ورودی‌های لایه‌ی l ، از ورودی‌ها که از لایه‌ی l-1 آمده
ضرب کردیم با اندیس ناول
اینجا activation function نداریم (خطی)

Subject:

Year: Month: Date:



حاصل تمرین خطا

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-K_1} \sum_{j=0}^{W-K_2} \left(\frac{\partial E}{\partial x^l_{i,j}} \times \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} \right) = \sum_{i=0}^{H-K_1} \sum_{j=0}^{W-K_2} \left(\delta^l_{i,j} \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} \right)$$

iterate کردن روی تمام خروجی‌های output که خروجی در $w^l_{m',n'}$ روی تمام خروجی‌های x^l تاثیر گذار بوده

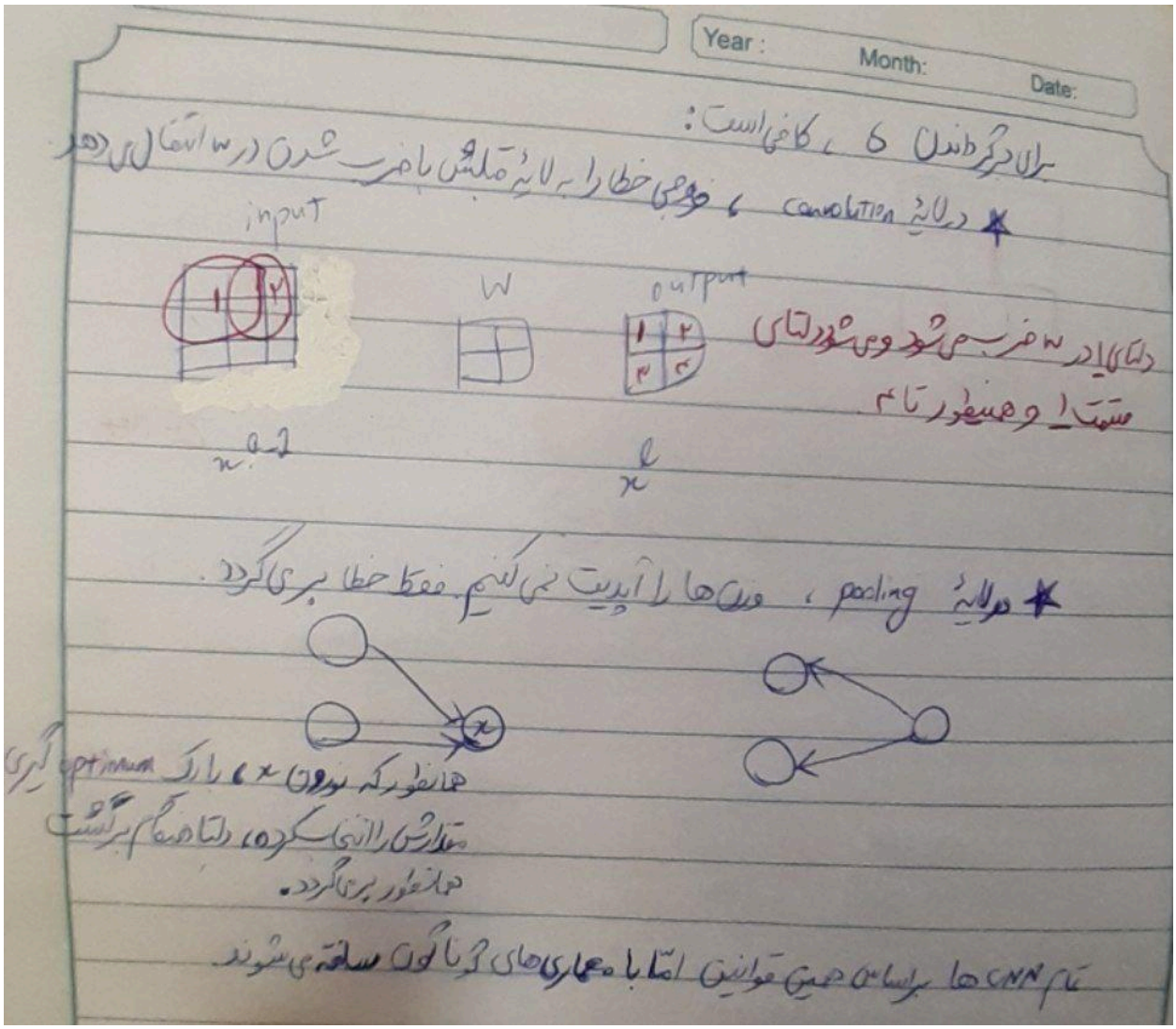
$$\frac{\partial x^l_{i,j}}{\partial w^l_{m,n}} = \frac{\partial}{\partial w^l_{m',n'}} \left(\sum_m \sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l \right)$$

$$= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{0,0} o^{l-1}_{i+0,j+0} + \dots + w^l_{m',n'} o^{l-1}_{m'+i,j+n'} + \dots + b^l \right)$$

$$= o^{l-1}_{m'+i,n'+j}$$

$$\Rightarrow \frac{\partial E}{\partial w^l_{m,n}} = \sum_{i=0}^{H-K_1} \sum_{j=0}^{W-K_2} \delta^l_{i,j} o^{l-1}_{i+m,j+n}$$

تمرین



سوال۳-

لایه های پولینگ سایز مسئله را کاهش میدهند و اطلاعات بی فایده را دور میریزند. همچنین، عمل scaling را انجام میدهند مثلا در max pooling، اطلاعات با اسکیل بالاتر را حفظ میکنند. این کار باعث میشود بار محاسباتی برای لایه های کانولوشن بعدی کمتر شود زیرا اطلاعات به نوعی فشرده شده است. تعداد پارامترهای مدل کمتر میشود و راحتتر یاد میگیرد. همچنین باعث میشود به موقعیت مکانی آبجکت های درون عکس،

invariant شود. همچنین فیچرها را میتوانیم با مکس پولینگ سلکت کنیم (فیچرها مهمتر) و اوریج پولینگ نیز میانگینی از همه اطلاعات را حفظ میکند.

نقاط منفی: باعث از دست رفتن بخشی از دیتا میشود. باعث افزوده شدن تعدادی هایپرپارامتر مثل سایز پولینگ و موقعیتشان در شبکه میشود.

در یک عکس عادی، اگر فیلترهای پولینگ را اعمال کنیم، در عکسهایی با بکگراند سفید، مکس پولینگ میتواند اشیای تاریک را کلا محو کند. و مین پولینگ در عکسهای با بکگراند مشکی، اشیای روشن را محو کند. اما در کل بیشتر از مکس پول استفاده میشود تا فیچرهای مهمتر از دست نروند.

استفاده بیش از حد از پولینگ نیز طبیعتا منجر به از دست رفتن دیتا میشود که شاید برای تسک مهم باشد.

سوال۴-

ترنسفر لرنینگ به ما امکان میدهد که از یک مدل از قبل اموزش داده شده استفاده کنیم تا تسک جدیدی را با فاین تون کردن مدل روی دیتاست جدید انجام دهیم. این کار باعث میشود بطور بهینه و بدون صرف انرژی زیاد، تسک را به انجام برسانیم. در واقع یک سری لایه اولیه با وزنهای شبکه‌ی pretrained فریز میشوند و فقط لایه های جدید اضافه شده، گرادیان را درون خود به عقب نشر میدهند.

از جمله این مدلها، alexnet برای ایمیج کلسیفیکیشن، vgg که فیچرهای عکسها را برای ما اکسترکت میکند، resnet برای ایمیج کلسیفیکیشن، ابجکت دیتکشن و سگمنتیشن، گوگل نت نیز برای ایمیج کلسیفیکیشن و ابجکت دیتکشن.

سوال۵-

-درست، وریفیکیشن یعنی یک چهره را با چهره سیو شده تطابق بدهیم تا بفهمیم آیا این فرد، شخص مورد نظر هست یا نه، یعنی خروجی یک باینری بله یا خیر است. اما رکوگنیشن یعنی تشخیص دهد که هویت این فرد کیست (از بین دیتابیسی بزرگ).

-نادرست، چون این تعداد دیتا ممکن است باعث شود شبکه به خوبی یاد نگیرد که مثلا به نورهای مختلف invariant شود. بهتر است از هر فرد با لیبل مشخص، چندین عکس با زوایا و نور متفاوت داده شود.

-درست، چون لایه های اولیه معمولا فیچرهای ساده تر را اکسترکت میکنند و لایه های بعدی فیچرهای پیچیده تر هستند و این را در فرایند دیکانولوشن هم میتوانیم ببینیم که هرچه جلوتر میرویم کرنل ها اشیای پییده تری را دیتکت میکنند.