مسئله‌ی ۱.

توضیح دهید که چطور می‌توان مقدار مناسب برای تعداد دسته‌ها را در یک الگوریتم یادگیری بدون نظارت[1] پیدا کرد.

در کل پیدا کردن تعداد cluster ها کار دشواری است. در ادمه چند رویکرد که به این منظور مورد استفاده قرار میگرد را بیان میکنیم.

۱. **Elbow** Method : این روش یک تکنیک محبوب برای تخمین تعداد بهینه خوشه ها است. این شامل ترسیم تعداد خوشه ها در برابر معیاری از تناسب مدل، مانند مجموع مجذور فاصله بین نقاط داده و مرکز خوشه آنها است. plot یک " Elbow " را تشکیل می دهد و تعداد خوشه ها در این نقطه انتخاب خوبی در نظر گرفته می شود. این نقطه مبادله بین پیچیدگی مدل و فشردگی خوشه ها را نشان می دهد. کاهش شدیدتر در اندازه گیری تناسب قبل از Elbow نشان دهنده انتخاب بهتر خوشه ها است. در واقع k-means را برای مقدیر مختلف اجرا میکند.

۲. **Silhouette** Analysis: تجزیه و تحلیل Silhouette کیفیت خوشه بندی را با محاسبه میانگین ضریب silhouette برای هر نقطه داده اندازه گیری می کند. ضریب silhouette نشان می دهد که یک نقطه داده چقدر در خوشه اختصاص داده شده خود در مقایسه با خوشه های دیگر قرار می گیرد. از ۱- تا ۱ متغیر است، جایی که مقادیر بالاتر نشان دهنده خوشه بندی بهتر است. با تغییر تعداد خوشه ها، می توان حداکثر امتیاز سیلوئت را شناسایی کرده و تعداد خوشه های مربوطه را انتخاب کرد.

۳. **Gap** Statistic: پراکندگی درون خوشه ای داده ها را برای مقادیر مختلف خوشه ها با پراکندگی مورد انتظار آن تحت توزیع های null reference مقایسه می کند. این به تعیین تعداد خوشه هایی کمک می کند که در آن گپ بین پراکندگی مشاهده شده و مورد انتظار بیشترین است. ایده این است که تعداد بهینه خوشه ها مربوط به نقطه ای است که آمار گپ به حداکثر می رسد.

۴. The **Sum of Squares**: یکی دیگر از روش‌های اعتبارسنجی خوشه‌بندی، انتخاب تعداد بهینه خوشه با به حداقل رساندن مجموع مربع‌های درون خوشه‌ای و به حداکثر رساندن مجموع مربع‌های بین خوشه‌ها (معیار میزان جدایی هر خوشه از بقیه).

مسئله‌ی ۲.

روشی ارائه کنید تا بتوان با آن تعداد رنگ‌های مختلف موجود در یک عکس را کاهش داد در حالی که همچنان محتوای عکس قابل تضخیص باشد. جزئیات اینکه چگونه به هر پیکسل رنگ جدید را اختصاص می‌دهید را در راه حل به‌صورت کامل توضیح دهید.

برای کاهش تعداد رنگ های مختلف در یک تصویر با حفظ قابلیت تشخیص محتوای آن، می توان از تکنیک های مختلفی استفاده نمود.

color quantization:

۱. بارگذاری تصویر: تصویری را که می خواهید پردازش کنید بخوانید و بارگذاری کنید.
۲. تعداد رنگ های مورد نظر را تعیین کنید: در مورد تعداد رنگ های مورد نظر که می خواهید تصویر پس از فرآیند کاهش داشته باشد، تصمیم بگیرید.
۳. جمع آوری پالت رنگ: رنگ های موجود در تصویر را با تکرار در هر پیکسل و ذخیره مقادیر RGB آنها در یک پالت استخراج کنید.

۴. اعمال خوشه‌بندی: از یک الگوریتم خوشه‌بندی مانند K-means یا نسخه اصلاح‌شده‌ای که به‌طور خاص برای کمی‌سازی رنگ طراحی شده است، استفاده کنید تا رنگ‌های موجود در پالت را خوشه‌بندی کنید. هر رنگ را به نزدیکترین مرکز خوشه آن اختصاص دهید.
۵. کاهش پالت رنگ: پس از خوشه بندی، مجموعه ای از رنگ های نماینده خواهید داشت. با انتخاب رنگ‌هایی که از نظر بصری مهم‌ترین رنگ‌ها هستند، اندازه پالت را به تعداد رنگ‌های دلخواه کاهش دهید. می توانید از روش های مختلفی مانند رتبه بندی رنگ ها بر اساس فراوانی وقوع یا اهمیت ادراکی آنها استفاده کنید.
۶. نقشه رنگ‌ها به پالت کاهش‌یافته: در هر پیکسل در تصویر تکرار کنید و رنگ اصلی آن را با نزدیک‌ترین رنگ از پالت کاهش‌یافته جایگزین کنید. نزدیک‌ترین رنگ را می‌توان با محاسبه فاصله اقلیدسی یا فاصله رنگ دیگری بین رنگ اصلی پیکسل و رنگ‌های موجود در پالت کاهش‌یافته تعیین کرد. نزدیک ترین رنگ را به پیکسل اختصاص دهید.
۷. ذخیره تصویر اصلاح شده: تصویر حاصل را با پالت رنگ کاهش یافته ذخیره کنید.

Median Cut: الگوریتم برش میانه یک روش محبوب برای کمی سازی رنگ است. با پارتیشن بندی بازگشتی فضای رنگی بر اساس توزیع رنگ ها در تصویر کار می کند. الگوریتم با کل فضای رنگ شروع می شود و به طور مکرر آن را در امتداد ابعاد با بیشترین واریانس تقسیم می کند تا به تعداد رنگ مورد نظر برسد. این روش تمایل به حفظ توزیع کلی رنگ دارد و می تواند نتایج بصری دلپذیری را ایجاد کند.

Neural Network-based Quantization: رویکردهای یادگیری عمیق نیز می توانند برای کوانتیزه کردن رنگ مورد استفاده قرار گیرند. با آموزش یک شبکه عصبی بر روی مجموعه داده های بزرگی از تصاویر، می توانید یک نقشه برداری از رنگ های اصلی به یک پالت رنگ کاهش یافته را بیاموزید. شبکه را می توان آموزش داد تا تفاوت رنگ بین تصاویر اصلی و کوانتیزه شده را به حداقل برساند و در عین حال محتوای بصری تصویر را حفظ کند.

Dithering: Dithering تکنیکی است که نویز بصری را برای شبیه سازی رنگ های اضافی فراتر از پالت رنگ کاهش یافته معرفی می کند. خطای کوانتیزاسیون را از پیکسل های مجاور توزیع می کند تا توهم رنگ های اضافی و گرادیان های صاف را ایجاد کند. الگوریتم های Dithering، مانند Dithering Floyd-Steinberg، را می توان پس از کاهش پالت رنگ برای بهبود کیفیت بصری و کاهش مصنوعات نوار رنگ اعمال کرد.

LeNet : برای تطبیق LeNet برای کمی سازی رنگ، باید تغییراتی را برای مدیریت کانال های رنگی در تصاویر ورودی انجام دهید. در اینجا یک طرح کلی از نحوه استفاده از LeNet برای کوانتیزه کردن رنگ آورده شده است:

آماده سازی مجموعه داده: مجموعه داده ای از تصاویر ورودی را برای آموزش آماده کنید. هر تصویر باید یک پیکسل از تصویر اصلی را نشان دهد و مقادیر رنگ آن ویژگی های ورودی شبکه خواهد بود.

معماری شبکه: لایه ورودی LeNet را برای پذیرش تصاویر رنگی تغییر دهید. لایه ورودی باید دارای سه کانال مربوط به کانال های رنگ قرمز، سبز و آبی باشد. ممکن است لازم باشد ابعاد لایه‌های بعدی را برای تغییر اندازه ورودی تنظیم کنید.

آموزش: LeNet اصلاح شده را در مجموعه داده خود با استفاده از یک تابع ضرر مناسب آموزش دهید. از آنجایی که کمی کردن رنگ یک کار بدون نظارت است، می‌توانید از یک تابع از دست دادن مانند میانگین مربعات خطا (MSE) برای به حداقل رساندن تفاوت بین رنگ پیکسل اصلی و مقدار کمی آن استفاده کنید.

Inference: پس از آموزش، می توانید از مدل LeNet آموزش دیده برای پیش بینی مقادیر رنگ کوانتیزه شده برای هر پیکسل در تصویر ورودی استفاده کنید. می‌توانید مدل را به‌صورت جداگانه روی هر پیکسل اعمال کنید یا از پنجره‌های کشویی برای پردازش وصله‌های تصویر بزرگ‌تر استفاده کنید.

نقشه برداری رنگ: مقادیر رنگی پیش بینی شده را به یک پالت رنگ کاهش یافته ترسیم کنید. نقشه برداری می تواند بر اساس نزدیک ترین همسایه یا رویکرد پیچیده تر مانند خوشه بندی رنگ های پیش بینی شده به نزدیک ترین رنگ ها در پالت کاهش یافته باشد.

بازسازی: مقادیر رنگ اصلی هر پیکسل را در تصویر ورودی با رنگ های کوانتیزه شده مربوطه از پالت کاهش یافته جایگزین کنید.

یک راهکار دیگر میتواند این باشد که روی یک عکس کرنل با اندازه مناسب و مقدار مناسب اعمال کنیم و گام های جلو رفتن کرنل را به اندازه طول آن قرار دهیم. با این کار به ازای هر قسمت n*n از تصویر فقط ک رنگ می ماند ولی لزوما تصویر واضح نمی ماند و بستگی به ابعاد کرنل دارد.

مسئله ۳.

$H_1 \rightarrow w_{11}\cdot x_1 + w_{21}\cdot x_2 + b_1\cdot 1 \Rightarrow 0{,}15\times 0{,}05 + 0{,}2\times 0{,}7 + 0{,}35\times 1 = 0{,}37775$

Sigmoid $\rightarrow \frac{1}{1+e^{-0{,}37775}} = 0{,}59327$

$H_2 \rightarrow 0{,}25\times 0{,}05 + 0{,}3\times 0{,}1 + 0{,}35\times 1 = 0{,}3925$

Sigmoid $\rightarrow \frac{1}{1+e^{-0{,}3925}} = 0{,}59688$

$y1 \rightarrow H_1\cdot u_{11} + H_2\cdot u_{21} + b_2\cdot 1 = 0{,}4\times 0{,}593 + 0{,}45\times 0{,}597 + 0{,}6\times 1 = 1{,}1059$

Sigmoid $\rightarrow \frac{1}{1+e^{-1{,}1059}} = 0{,}751$

$y2 \rightarrow 0{,}593\times 0{,}5 + 0{,}55\times 0{,}597 + 0{,}6\times 1 = 1{,}225$

Sigmoid $\rightarrow \frac{1}{1+e^{-1{,}1225}} = 0{,}773$

Error $\rightarrow$ 1: $1/2\,(0{,}1 - 0{,}751)^2 = 0{,}212$ 2: $1/2\,(0{,}99 - 0{,}773)^2 = 0{,}024$ $\xrightarrow{\text{Sum}}$ 0,236

$Error_w = \frac{\partial E_T}{\partial w} \rightarrow$ قانون زنجیری

$\frac{\partial E_T}{\partial y_1} \Rightarrow -(0{,}1 - 0{,}751) = 0{,}651 \quad \frac{\partial y_1}{\partial y_1} = 0{,}751\times(1 - 0{,}751) = 0{,}187 \quad \frac{\partial y_1}{\partial u_{11}} = H_1 = 0{,}593$

$Error\ u_{11} = 0{,}651\times 0{,}187\times 0{,}593 = 0{,}0721$

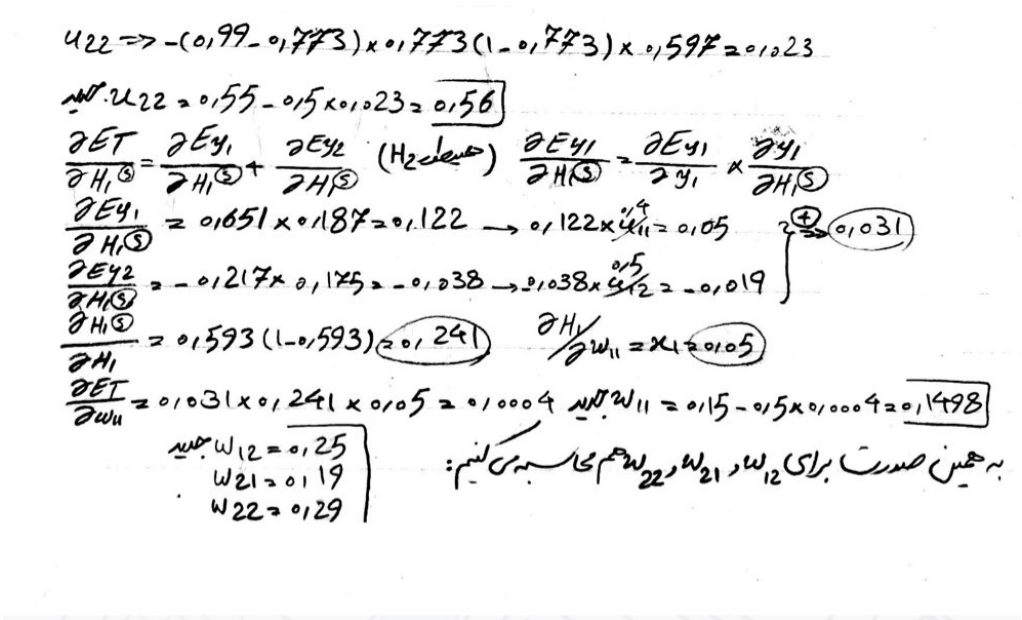
$u_{11} = u_{11} - 0{,}5\frac{\partial E_T}{\partial u_{11}} \Rightarrow 0{,}4 - 0{,}5\times 0{,}0721 = 0{,}36$ (مقدار جدید)

$u_{12} \Rightarrow -(0{,}99 - 0{,}773)\times 0{,}773(1 - 0{,}773)\times 0{,}593 = 0{,}023$

مقدار جدید $u_{12} = 0{,}5 - 0{,}5\times 0{,}023 = 0{,}51$

$u_{21} \Rightarrow -(0{,}1 - 0{,}751)\times 0{,}751(1 - 0{,}751)\times 0{,}597 = 0{,}072$

مقدار جدید $u_{21} = 0{,}45 - 0{,}5\times 0{,}072 = 0{,}41$



مسئله ۴.

نحوه کارکرد شبکه عصبی پیچشی را توضیح دهید:

شبکه عصبی کانولوشنی (CNN) نوعی مدل یادگیری عمیق است که به طور خاص برای پردازش داده‌های شبکه‌مانند ساختاریافته، مانند تصاویر، طراحی شده است. CNN ها به طور گسترده در وظایف بینایی کامپیوتری، از جمله طبقه بندی تصویر، تشخیص اشیا و تقسیم بندی تصویر استفاده می شوند.

نحوه عملکرد CNN:

۱. لایه کانولوشنال: اولین لایه در CNN معمولاً یک لایه کانولوشن است. این شامل چندین فیلتر (kernels) است که روی تصویر ورودی کانولو میشود. هر فیلتر یک عملیات محصول نقطه ای را بین وزن خود و یک میدان پذیرنده کوچک (patch) از تصویر ورودی انجام می دهد. نتیجه یک feature map است که الگوهای محلی یا ویژگی های مکانی را در تصویر ثبت می کند.

۲. تابع فعال سازی: پس از عملیات کانولوشن، یک تابع فعال سازی مانند ReLU (واحد خطی اصلاح شده) به صورت عنصری برای معرفی غیر خطی اعمال می شود. این به شبکه اجازه می دهد تا روابط پیچیده بین ویژگی ها را بیاموزد.

۳. Pooling Layer: یک لایه Pooling از لایه Convolutional پیروی می کند و ابعاد فضایی feature map هارا کاهش می دهد. عملیات پولینگ، مانند max pooling یا average pooling ، یک منطقه از feature map را با گرفتن حداکثر یا میانگین مقدار در آن منطقه خلاصه می کند. به کاهش پیچیدگی محاسباتی کمک می کند و شبکه را نسبت به تغییرات فضایی کوچک تغییر نمی دهد.

۴. Fully Connected Layers: feature mapها که توسط لایه‌های کانولوشنال و پولینگ تولید می‌شوند، سپس صاف می‌شوند و از یک یا چند لایه کاملاً متصل عبور می‌کنند. این لایه‌ها مشابه لایه‌های شبکه‌های عصبی سنتی هستند که در آن هر نورون به هر نورون در لایه قبلی متصل است. لایه‌های کاملاً متصل ویژگی‌های سطح بالا را می‌گیرند و با ترکیب ویژگی‌های سطح پایین، الگوهای پیچیده را یاد می‌گیرند.

۵. لایه خروجی: آخرین لایه fully connected layer لایه خروجی است که پیش بینی ها یا طبقه بندی های نهایی را تولید می کند. تعداد نورون ها در این لایه به وظیفه خاص بستگی دارد. به عنوان مثال، در طبقه‌بندی تصویر، هر نورون مربوط به یک برچسب کلاس است و خروجی از یک تابع فعال‌سازی softmax برای تولید احتمالات کلاس عبور می‌کند.

۶. آموزش: CNN ها با استفاده از داده های برچسب گذاری شده از طریق backpropagation آموزش داده می شوند. در طول تمرین، شبکه وزن‌ها و بایاس‌های خود را به طور مکرر تنظیم می‌کند تا با مقایسه خروجی‌های پیش‌بینی‌شده با برچسب‌های واقعی، یک loss function (مانند cross-entropy loss) را به حداقل برساند. این معمولاً با استفاده از الگوریتم‌های بهینه‌سازی مانند نزول گرادیان تصادفی (SGD) یا انواع آن انجام می‌شود.

۷. Inference: هنگامی که CNN آموزش داده شد، می توان از آن برای استنباط بر روی داده های جدید و دیده نشده استفاده کرد. تصاویر ورودی از طریق شبکه ارسال می‌شوند و خروجی‌های لایه نهایی، پیش‌بینی‌ها یا احتمالات را برای کلاس‌های مختلف نشان می‌دهند.

CNN ها از ساختار سلسله مراتبی داده ها برای یادگیری خودکار ویژگی های مرتبط در سطوح مختلف انتزاع استفاده می کنند. این ویژگی، همراه با اشتراک‌گذاری وزن و میدان‌های دریافت محلی، CNNها را قادر می‌سازد تا به طور موثر ویژگی‌های تصاویر را استخراج و طبقه‌بندی کنند.

مسئله ۵.

Adam (Adaptive Moment Estimation) یک الگوریتم بهینه سازی است که معمولاً در شبکه های عصبی برای به روز رسانی پارامترها در طول فرآیند آموزش استفاده می شود. این روش ترکیبی از مزایای دو روش بهینه سازی محبوب دیگر، یعنی AdaGrad و RMSprop است. Adam به دلیل کارایی و اثربخشی خود در آموزش شبکه های عصبی عمیق شناخته شده است.

ایده کلیدی پشت آدام تنظیم تطبیقی نرخ یادگیری برای هر پارامتر در شبکه بر اساس تخمین گشتاور مرتبه اول (میانگین) و لحظه مرتبه دوم (واریانس بدون مرکز) گرادیان ها است.

نحوه کار:

۱. مقداردهی اولیه: برای هر پارامتر در شبکه، متغیرهای لحظه اول و دوم را که به ترتیب با "m" و "v" مشخص می شوند، مقداردهی کنید. این متغیرها به صورت بردارهای صفر مقداردهی اولیه می شوند.

۲. محاسبه گرادیان ها: گرادیان پارامترهای شبکه را با توجه به loss functionبا استفاده از backpropagation محاسبه کنید.

۳. برآورد لحظه اول را به روز کنید: تخمین لحظه اول 'm' را با استفاده از واپاشی نمایی به روز کنید. این کار برای پیگیری میانگین گرادیان ها انجام می شود.

m = beta1 * m + (1 - beta1) * gradient

در اینجا، beta1 یک فراپارامتر بین ۰ و ۱ است (معمولاً ۰.۹ تنظیم می شود)، که نرخ فروپاشی تخمین لحظه اول را کنترل می کند.

۴. به روز رسانی تخمین لحظه دوم: تخمین لحظه دوم 'v' را با استفاده از واپاشی نمایی به روز کنید. این تخمین واریانس بدون مرکز گرادیان ها را دنبال می کند. قانون به روز رسانی برای 'v' به شرح زیر است:

v = beta2 * v + (1 - beta2) * gradient^2

در اینجا، beta2 یکی دیگر از هایپرپارامترهای بین ۰ و ۱ است (معمولاً ۰.۹۹۹ تنظیم می شود)، که نرخ فروپاشی تخمین لحظه دوم را کنترل می کند.

۵. تصحیح سوگیری: از آنجایی که لحظه های 'm' و 'v' به عنوان بردارهای صفر مقدار دهی اولیه می شوند، به ویژه در مراحل اولیه آموزش به سمت صفر سوگیری می کنند. برای اصلاح این سوگیری، آدام یک مرحله تصحیح سوگیری را اعمال می کند:

m_hat = m / (1 - beta1^t)

v_hat = v / (1 - beta2^t)

در اینجا، 't' مرحله زمانی را نشان می دهد که از ۱ شروع می شود و پس از هر به روز رسانی پارامتر افزایش می یابد.

۶. به روز رسانی پارامترها: در نهایت، پارامترها با استفاده از لحظه های محاسبه شده به روز می شوند. قانون به روز رسانی برای هر پارامتر به شرح زیر است:

parameter = parameter - (learning_rate * m_hat) / (sqrt(v_hat) + epsilon)

در اینجا، 'learning_rate' فراپارامتر نرخ یادگیری است، و 'epsilon' یک ثابت کوچک است (به عنوان مثال، e-8۱) که برای ثبات عددی برای جلوگیری از تقسیم بر صفر اضافه شده است.

با تطبیق نرخ یادگیری بر اساس تخمین لحظه های اول و دوم، آدام می تواند به طور خودکار نرخ یادگیری را برای هر پارامتر به صورت جداگانه تنظیم کند و منجر به همگرایی کارآمد در طول آموزش شود. ماهیت تطبیقی Adam به آن اجازه می دهد تا انواع مختلفی از گرادیان ها، از جمله گرادیان های پراکنده را مدیریت کند، و به طور گسترده ای به عنوان یک بهینه ساز قابل اعتماد برای معماری های مختلف شبکه عصبی مورد استفاده قرار گرفته است.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1$^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize 2$^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)