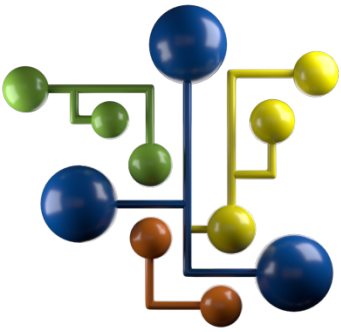


Data Science Academy

www.datascienceacademy.com.br

Processamento de Linguagem Natural

Como Funcionam as CNNs?

Redes Neurais Convolucionais parecem uma combinação estranha de biologia e matemática com um pouco de Ciência da Computação e essa categoria de redes neurais tem sido uma das inovações mais influentes no campo da Visão Computacional. O marco destas redes neurais foi em 2012, o primeiro ano em que cresceram de forma proeminente, quando Alex Krizhevsky as usou para conquistar a competição ImageNet daquele ano (basicamente, as Olimpíadas anuais de Visão Computacional), reduzindo o registro de erros de classificação de 26% para 15%, uma melhora considerável. Desde então, uma série de empresas têm utilizado o aprendizado profundo no centro de seus serviços. O Facebook usa redes neurais convolucionais para seus algoritmos de marcação automática, o Google para sua pesquisa de fotos, Amazon para suas recomendações de produtos, Pinterest para a personalização do feed de notícias e Instagram para sua infraestrutura de pesquisa. Alex Krizhevsky fazia parte do grupo de Geoffrey Hinton da Universidade de Toronto no Canadá e foi contratado pelo Google depois de vencer a competição ImageNet.

O clássico e, provavelmente o mais popular, caso de uso dessas redes é o processamento de imagens. Vamos dar uma olhada em como usar estas CNNs para esta atividade.

O Problema do Espaço

A classificação de uma imagem é a tarefa de receber uma imagem de entrada e produzir uma classe (um gato, cachorro, etc.) ou uma probabilidade de classes que melhor descrevem a imagem. Para os seres humanos, esta tarefa de reconhecimento é uma das primeiras habilidades que aprendemos desde o momento em que nascemos e de forma natural, sem esforço quando já estamos adultos. Sem ter que pensar duas vezes, podemos identificar de forma rápida e perfeita o ambiente em que estamos e os objetos que nos cercam. Quando vemos uma imagem ou apenas quando olhamos para o mundo que nos rodeia, na maioria das vezes somos capazes de caracterizar imediatamente a cena e dar a cada objeto um rótulo, tudo sem perceber conscientemente. Essas habilidades de poder reconhecer rapidamente os padrões, generalizar do conhecimento prévio e se adaptar a diferentes ambientes de imagem são aqueles que não compartilhamos com outras máquinas. Não ainda!

What We See

What Computers See

Entradas e Saídas

Quando um computador vê uma imagem (recebe uma imagem como entrada), verá uma série de valores de pixels. Dependendo da resolução e do tamanho da imagem, verá uma matriz de 32 x 32 x 3 (o 3 refere-se a valores RGB). Apenas para guiar nosso ponto de partida, digamos que temos uma imagem colorida em formato JPG e seu tamanho é 480 x 480. A matriz representativa será 480 x 480 x 3. Cada um desses números recebe um valor de 0 a 255 que descreve a intensidade do pixel nesse ponto. Esses números, embora sem sentido para nós quando executamos a classificação da imagem, são as únicas entradas disponíveis para o computador. A ideia é que você forneça ao computador essa série de números que vão produzir outros números que descrevem a probabilidade de a imagem ser uma determinada classe (0.80 para gato, 0.15 para cachorro, 0.05 para pássaro, etc.).

O que queremos que o Computador Faça?

Agora que conhecemos o problema, bem como as entradas e saídas, pensemos em como abordar isso. O que queremos que o computador faça é poder diferenciar entre todas as imagens que são fornecidas e descobrir as características únicas que fazem um cão um cachorro ou que fazem um gato um gato. Este é o processo que se passa também em nossas mentes subconscientemente. Quando olhamos para uma foto de um cão, podemos classificá-lo como tal, se a imagem tiver características identificáveis, como patas ou orelhas. De forma semelhante, o computador pode realizar a classificação da imagem procurando recursos de baixo nível, como bordas e curvas, e, em seguida, construindo conceitos mais abstratos através de uma série de camadas convolutivas. Esta é uma visão geral do que faz uma CNN. Vamos entrar em detalhes.

Conexão Biológica

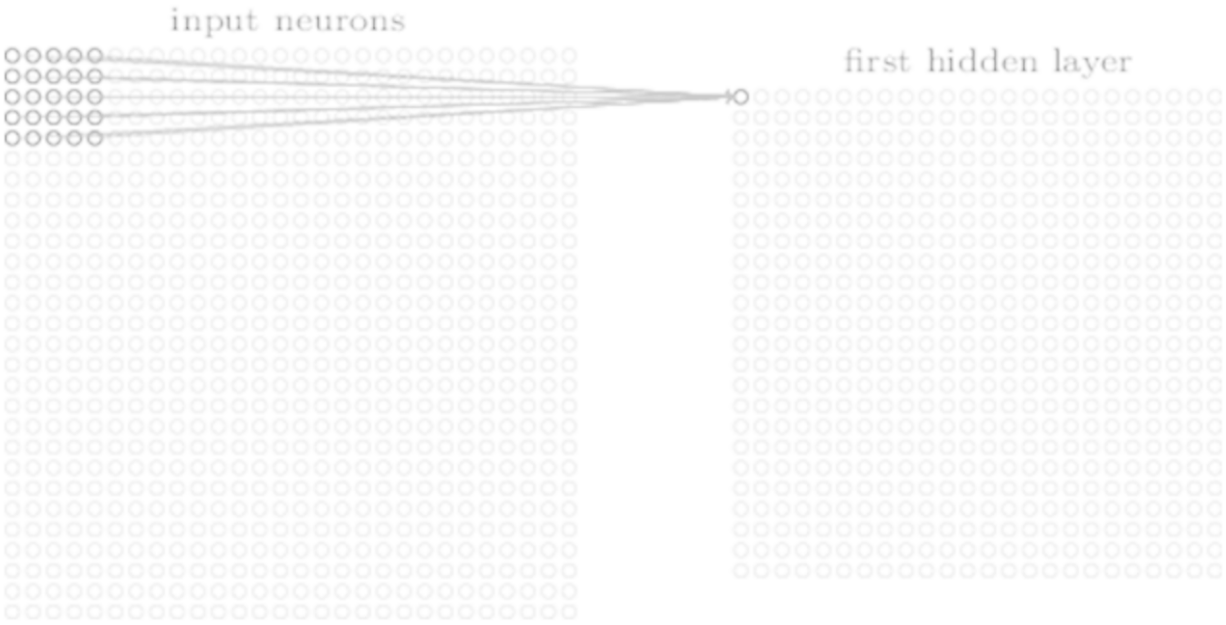
Mas, primeiro, um pouco de background. Quando você ouviu falar do termo redes neurais convolucionais, você pode ter pensado em algo relacionado à neurociência ou à biologia, e você teria razão. As CNNs têm uma inspiração biológica do córtex visual. O córtex visual possui pequenas regiões de células que são sensíveis a regiões específicas do campo visual. Essa ideia foi ampliada por um experimento fascinante de Hubel e Wiesel em 1962, onde eles mostraram que algumas células neuronais individuais no cérebro responderam (ou dispararam) apenas na presença de bordas de determinada orientação. Por exemplo, alguns neurônios disparados quando expostos a bordas verticais e alguns quando exibidos a bordas horizontais ou diagonais. Hubel e Wiesel descobriram que todos esses neurônios estavam organizados em uma arquitetura em coluna e que, juntos, conseguiram produzir percepção visual. Essa ideia de componentes especializados dentro de um sistema com tarefas específicas (as células neuronais no córtex visual à procura de características específicas) é aquela que as máquinas também usam e é a base das CNNs.

Estrutura

Voltemos para os detalhes. Uma visão geral mais detalhada do que as CNNs fazem seria a seguinte: você insere uma imagem como input, passando por uma série de camadas convolucionais, não-lineares, de pooling (downsampling) e totalmente conectadas, e obtém uma saída. Como dissemos anteriormente, a saída pode ser uma única classe ou uma probabilidade de classes que melhor descrevem a imagem. Agora, a parte mais difícil é entender o que cada uma dessas camadas faz. Então, vamos entrar no mais importante.

Primeira Camada - Parte de Matemática

A primeira camada em uma CNN é sempre uma camada convolucional. Como mencionamos anteriormente, a entrada pode ser uma matriz de 32 x 32 x 3 de valores de pixels. Agora, a melhor maneira de explicar uma camada convolucional é imaginar uma lanterna que está brilhando no canto superior esquerdo da imagem. Digamos que a luz que esta lanterna brilha abrange uma área de 5 x 5. E agora, vamos imaginar essa lanterna deslizando em todas as áreas da imagem de entrada. Em termos de aprendizagem de máquina, esta lanterna é chamada de **filtro** (ou às vezes referido como um neurônio ou um kernel) e a região que está brilhando é chamada de **campo receptivo**. Este filtro também é uma matriz de números (os números são chamados de pesos ou parâmetros). Uma observação muito importante é que a profundidade deste filtro deve ser a mesma que a profundidade da entrada (isso garante que a matemática funcione), então as dimensões deste filtro são 5 x 5 x 3. Agora, vamos considerar a primeira posição em que o filtro se encontra, por exemplo. Seria o canto superior esquerdo. À medida que o filtro está deslizando, ou "convolvendo", em torno da imagem de entrada, ele está multiplicando os valores no filtro com os valores de pixel originais da imagem (também conhecido como multiplicação de elementos de computação). Essas multiplicações são somadas (matematicamente falando, isso seria 75 multiplicações no total). Então agora você tem um único número. Lembre-se, este número é apenas representativo de quando o filtro está no canto superior esquerdo da imagem. Agora, repetimos esse processo para cada local no volume de entrada (o próximo passo seria mover o filtro para a direita por 1 unidade, depois para a direita novamente em 1, e assim por diante). Toda localização única no volume de entrada produz um número. Depois de deslizar o filtro sobre todos os locais, você descobrirá que o que lhe resta é uma matriz de 28 x 28 x 1, que chamamos de mapa de ativação ou mapa de recursos. A razão pela qual você obtém uma matriz de 28 x 28 é que existem 784 locais diferentes que um filtro de 5 x 5 pode caber em uma imagem de entrada de 32 x 32. Esses 784 números são mapeados para uma matriz de 28 x 28.



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Primeira Camada - Perspectiva de alto nível

No entanto, vamos falar sobre o que esta convolução está realmente fazendo a partir de um alto nível. Cada um desses filtros pode ser considerado como identificador de recursos. Quando digo características, estou falando sobre coisas como bordas retas, cores simples e curvas. Pense nas características mais simples que todas as imagens têm em comum entre si. Digamos que nosso primeiro filtro é 7 x 7 x 3 e será um detector de curva. (Nesta seção, vamos ignorar o fato de que o filtro é de 3 unidades de profundidade e apenas considerar a fatia de profundidade superior do filtro e a imagem, por simplicidade). Como um detector de curva, o filtro terá uma estrutura de pixels que terá altos valores numéricos ao longo da área que é uma forma de uma curva (Lembre-se, esses filtros que estamos falando como apenas números!).

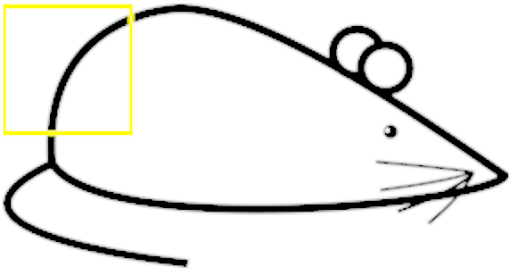
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Visualization of a curve detector filter

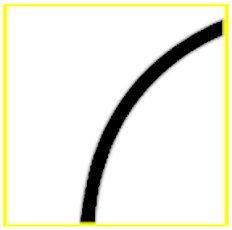
Agora, vamos voltar a visualizar isso matematicamente. Quando temos esse filtro no canto superior esquerdo do volume de entrada, teremos uma multiplicação entre os valores de filtro e pixels naquela região. Agora vamos dar um exemplo de uma imagem que queremos classificar, e vamos colocar o nosso filtro no canto superior esquerdo.

Original image



Visualization of the filter on the image

Lembre-se, o que temos a fazer é multiplicar os valores no filtro com os valores de pixel originais da imagem.



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

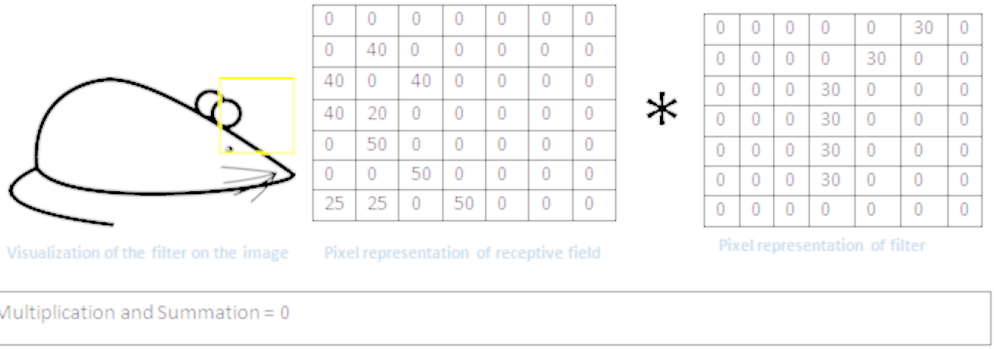
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

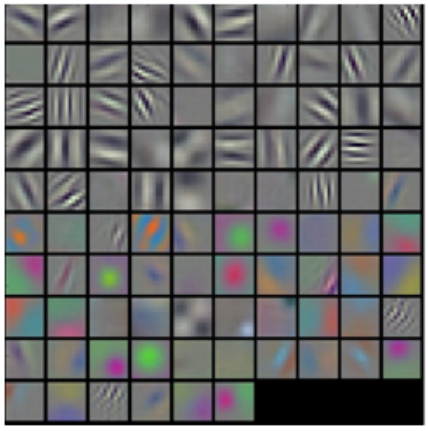
Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Basicamente, na imagem de entrada, se houver uma forma que geralmente se assemelhe à curva que este filtro representa, então todas as multiplicações somadas juntas resultarão em um grande valor! Agora vamos ver o que acontece quando movemos nosso filtro.



O valor é muito menor! Isso ocorre porque não havia nada na seção de imagem que respondeu ao filtro do detector de curva. Lembre-se, a saída desta camada convolucional é um mapa de ativação. Assim, no caso simples de uma convolução de um filtro (e se esse filtro for um detector de curva), o mapa de ativação mostrará as áreas em que há principalmente chances de serem curvas na imagem. Neste exemplo, o valor superior esquerdo do nosso mapa de ativação de 28 x 28 x 1 será 6600. Esse valor alto significa que é provável que exista algum tipo de curva no volume de entrada que fez com que o filtro fosse ativado. O valor superior direito em nosso mapa de ativação será 0 porque não houve nada no volume de entrada que fez com que o filtro fosse ativado (ou em outras palavras, não havia uma curva na região da imagem original). Lembre-se, isso é apenas para um filtro. Este é um filtro que vai detectar linhas que se curvam para fora e para a direita. Podemos ter outros filtros para linhas que se curvam para a esquerda ou para bordas retas. Quanto mais filtros, maior a profundidade do mapa de ativação e mais informações sobre o volume de entrada.

Obs: O filtro que descrevemos acima foi simplista para o propósito principal de descrever a matemática que ocorre durante uma convolução. Na imagem abaixo, você verá alguns exemplos de visualizações reais dos filtros da primeira camada de convolução da rede treinada. No entanto, o principal argumento permanece o mesmo. Os filtros na primeira camada convertem-se em torno da imagem de entrada e "ativam" (ou calculam valores altos) quando o recurso específico que está procurando está no volume de entrada.



Visualizations of filters

Indo Mais Fundo Pela Rede

Em uma arquitetura de rede neural *convencional* tradicional, existem outras camadas que são intercaladas com as camadas convolucionais (criando assim um modelo convolucional). Em um sentido geral, essas camadas fornecem não-linearidades e preservação de dimensões que ajudam a melhorar a robustez da rede e controlar o overfitting. Uma arquitetura clássica da CNN seria assim:

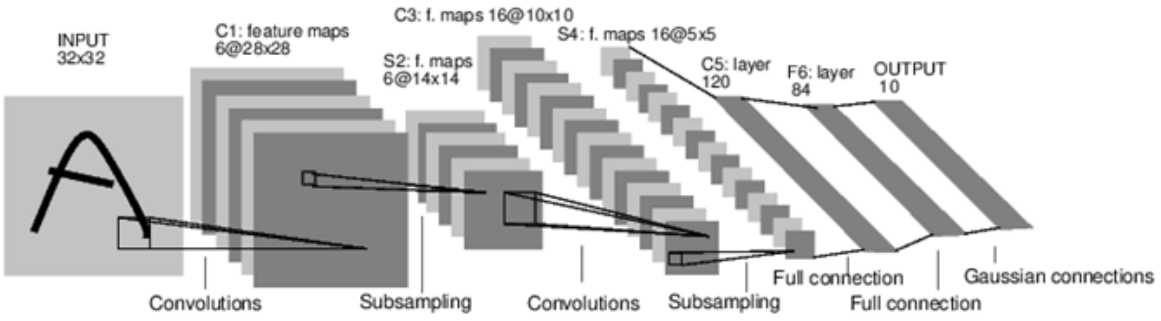
```
Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected
```

A última camada é importante e vamos entrar em detalhes daqui a pouco. Vamos dar um passo atrás e rever o que aprendemos até agora. Nós conversamos sobre o que os filtros da primeira camada convolucional são projetados para detectar. Eles detectam características de baixo nível, como bordas e curvas. Como se poderia imaginar, para prever se uma imagem é um tipo de objeto, precisamos que a rede possa reconhecer recursos de nível superior, como mãos ou patas ou ouvidos. Então, vamos pensar sobre o que é a saída da rede após a primeira camada convolucional. Seria um volume de 28 x 28 x 3 (assumindo que usamos três filtros de 5 x 5 x 3). Quando passamos por outra camada convolucional, a saída da primeira camada convolucional torna-se a entrada da segunda camada convolucional. Agora, isso é um pouco mais difícil de visualizar. Quando estávamos falando sobre a primeira camada, a entrada era apenas a imagem original. No entanto, quando estamos falando sobre a segunda camada convolucional, a entrada é o mapa de ativação que resulta da primeira camada. Então, cada camada da entrada está basicamente descrevendo os locais na imagem original para onde determinados recursos de baixo nível aparecem. Agora, quando você aplica um conjunto de filtros em cima disso (passando através da segunda camada convolucional), a saída será a ativação que representa recursos de nível superior. Os tipos dessas características podem ser semicírculos (combinação de uma curva e borda reta) ou quadrados (combinação de várias arestas retas). À medida que você atravessa a rede e passa por mais camadas convolucionais, você obtém mapas de ativação que representam características cada vez mais complexas. Ao final da rede, você pode ter alguns filtros que se ativam quando há caligrafia na imagem, filtros que se ativam quando veem objetos cor-de-rosa, etc. Outra coisa interessante a notar é que, à medida que você aprofunda a rede, os filtros começam a ter um campo receptivo maior e maior, o que significa que eles são capazes de considerar informações de uma área maior do volume de entrada original (em outras palavras eles são mais sensíveis a uma região maior de espaço de pixels). Veja esse vídeo no Youtube, com uma demonstração do que ocorre em cada camada de uma CNN:

Deep Visualization Toolbox: https://www.youtube.com/watch?v=AgkfIQ4IGaM

Camada Totalmente Conectada

Agora que podemos detectar esses recursos de alto nível, podemos anexar uma camada totalmente conectada ao final da rede. Esta camada basicamente possui um volume de entrada (qualquer que seja a saída da camada de convolução ou ReLU ou camada de pool que precede) e produzir um vetor N dimensional onde N é o número de classes que o programa deve escolher. Por exemplo, se você quisesse um programa de classificação de dígitos, N seria 10, pois há 10 dígitos. Cada número neste vetor N dimensional representa a probabilidade de uma determinada classe. Por exemplo, se o vetor resultante para um programa de classificação de dígitos é [0 .1 .1 .75. 0 0 0 0 0 .05], isso representa uma probabilidade de 10% de que a imagem é o dígito 1, 10% de probabilidade que a imagem é o dígito 2, 75% de que a imagem é um 3 e uma probabilidade de 5% de que a imagem é um 9 (Obs: existem outras maneiras de representar a saída, mas estamos apenas mostrando a abordagem softmax). A maneira como esta camada totalmente conectada funciona é que ela examina a saída da camada anterior (que, como lembramos, deve representar os mapas de ativação de recursos de alto nível) e determina quais características mais se correlacionam a uma determinada classe. Por exemplo, se o programa prevê que alguma imagem é um cão, ele terá valores elevados nos mapas de ativação que representam características de alto nível, como uma pata ou uma orelha, etc. Da mesma forma, se o programa prevê que alguma imagem seja um pássaro, terá valores elevados nos mapas de ativação que representam características de alto nível, como asas ou bico, etc. Basicamente, uma camada totalmente conectada examina quais recursos de alto nível se correlacionam mais fortemente a uma determinada classe e tem pesos específicos para que, quando você compute os produtos entre os pesos e a camada anterior, obtenha as probabilidades corretas para as diferentes classes.



A Full Convolutional Neural Network (LeNet)

Treinamento – Com Isso Realmente Funciona?

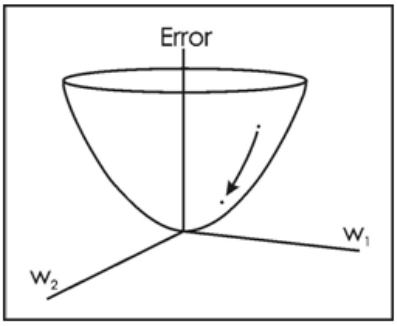
Este é o único aspecto das redes neurais que nós propositadamente ainda não mencionamos e provavelmente é a parte mais importante. É provável que você tenha muitas perguntas sobre o que leu até aqui. Como os filtros na primeira camada de convolução sabem procurar bordas e curvas? Como a camada totalmente conectada conhece os mapas de ativação a serem vistos? Como os filtros em cada camada sabem quais os valores devem ser usados? A maneira como o computador é capaz de ajustar seus valores de filtro (ou pesos) é através de um processo de treinamento chamado backpropagation, que estudamos bastante ao longo dos capítulos anteriores.

Antes de entrar em backpropagation, primeiro devemos dar um passo atrás e falar sobre o que uma rede neural precisa para funcionar. No momento em que todos nascemos, nossas mentes estão frescas. Nós não sabemos o que é um gato ou cachorro ou pássaro. De forma semelhante, antes do início do treinamento de uma CNN, os pesos ou os valores do filtro são randomizados, pois a rede não sabe que valores são ideais antes de começar o treinamento. Os filtros não sabem procurar bordas e curvas. Os filtros nas camadas mais altas não sabem procurar por patas e bicos. Conforme envelhecemos, nossos pais e professores nos mostraram imagens diferentes e nos dão um rótulo correspondente. Essa ideia de ser dada uma imagem e um rótulo é o processo de treinamento que as CNNs usam (aprendizagem supervisionada).

Portanto, a pós-propagação pode ser separada em 4 seções distintas, a passagem para a frente, a função de perda, a passagem para trás e a atualização de peso. Durante a passagem para a frente, você recebe uma imagem de treinamento que, como lembramos, pode ser uma matriz de 32 x 32 x 3 e passamos por toda a rede. No nosso primeiro exemplo de treinamento, uma vez que todos os pesos ou valores de filtro foram inicializados aleatoriamente, a saída provavelmente será algo como [.1 .1 .1 .1 .1 .1 .1 .1 .1 .1], basicamente uma saída que não dá preferência a qualquer número em particular. A rede, com seus pesos atuais, não é capaz de procurar esses recursos de baixo nível ou, portanto, não é capaz de fazer qualquer conclusão razoável sobre qual classificação final utilizar. Agora é a hora da função de perda fazer seu trabalho. Lembre-se que o que estamos usando dados para treinamento da rede. Esses dados possuem uma imagem e um rótulo. Digamos, por exemplo, que a primeira imagem de treinamento inserida era uma 3. O rótulo da imagem seria [0 0 0 1 0 0 0 0 0 0]. Uma função de perda vai olhar para a previsão da rede e comparar com o rótulo original, a fim de calcular o erro de previsão. Uma função de perda pode ser definida de muitas maneiras diferentes, mas uma opção comum é o MSE (erro quadrático médio), que é ½ vezes (real - previsto) ao quadrado. Ou para ser mais claro, uma fórmula como essa:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Digamos que a variável L é igual a esse valor. Como você pode imaginar, a perda será extremamente alta para as primeiras imagens de treinamento. Agora, pensemos nisso intuitivamente. Queremos chegar a um ponto em que o rótulo previsto (saída da ConvNet) é o mesmo que o rótulo de treinamento (isso significa que nossa rede obteve sua predição correta). Para chegar lá, queremos minimizar a perda. Visualizando isso como apenas um problema de otimização no cálculo, queremos descobrir quais entradas (pesos no nosso caso) contribuíram mais diretamente para a perda (ou erro) da rede.



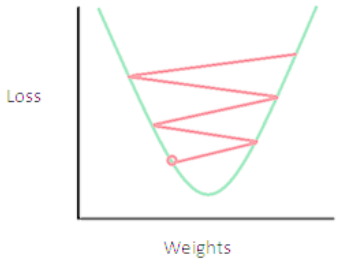
One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

Este é o equivalente matemático de dL/dW onde W são os pesos em uma camada específica. Agora, o que queremos fazer é executar uma passagem backward através da rede, que é determinar quais pesos contribuíram mais para a perda e encontrar maneiras de ajustá-los para que a perda diminua. Uma vez que calculamos esta derivada, então vamos para o último passo, que é a atualização de peso. Aqui é onde nós recebemos todos os pesos dos filtros e atualizamos para que eles mudem na direção oposta do gradiente.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

A taxa de aprendizado nesta fórmula é um parâmetro escolhido pelo desenvolvedor. Uma alta taxa de aprendizado significa que etapas maiores são tomadas nas atualizações de peso e, portanto, pode demorar menos tempo para que o modelo possa convergir em um conjunto ideal de pesos. No entanto, uma taxa de aprendizagem que é muito alta pode resultar em saltos que são muito grandes e não precisos o suficiente para alcançar o ponto ideal.



O processo de passagem para a frente, função de perda, passagem para trás e atualização de parâmetros é uma iteração de treinamento. O programa irá repetir este processo um número fixo de iterações para cada conjunto de imagens de treinamento (geralmente chamado de lote). Depois de terminar a atualização do parâmetro no último exemplo de treinamento, esperamos que a rede esteja treinada o suficiente para que os pesos das camadas sejam sintonizados corretamente.

Teste

Finalmente, para ver se a nossa CNN funciona ou não, temos um conjunto diferente de imagens e rótulos (dados de teste) e passamos as imagens pelo modelo CNN criado. Comparamos as saídas com os dados originais e então avaliamos se nossa rede funciona ou não!

Como as Empresas usam CNNs

Dados, dados, dados. As empresas que possuem esse precioso recurso são as que têm uma vantagem sobre seus competidores. Quanto mais dados de treinamento você puder entregar a uma rede, mais informações de treinamento você pode adquirir, mais atualizações de peso você pode fazer, e melhor ajustado será seu modelo quando vai para a produção. O Facebook e Instagram podem usar todas as fotos dos bilhões de usuários que atualmente possuem, o Pinterest pode usar informações dos 50 bilhões de “pins” que estão em seu site, o Google pode usar dados de pesquisa e a Amazon pode usar dados de milhões de produtos que são comprados todos os dias. E agora você conhece a magia por trás de como eles fornecem suas soluções.

Mas não é apenas em redes sociais ou sites de comércio eletrônico que as CNNs vem sendo usadas. Em Medicina, as CNNs vem sendo usadas para ajudar a detectar doenças a partir de raios-x ou fotos de pacientes. Empresas de alimentos vem usando CNNs para criar aplicações que reconhecem imagens de alimentos e saber quando estão cozidos ou prontos para consumo. Drones podem sobrevoar estoques de produtos e detectar falhas ou problemas, ou mesmo fazer a leitura de código de barras. Empresas de trânsito podem detectar motoristas que estejam com comportamento estranho através do reconhecimento de imagens.

E isso é só o começo!

Referência:

Visualizing and Understanding Convolutional Networks
http://www.matthewzeiler.com/wp-content/uploads/2017/07/arxive2013.pdf