Classificação ImageNet com Redes Neurais Convolucionais

Alex Krizhevsky[1], **Ilya Sutskever**[1],**Geoffrey E. Hinton**[1]

[1]

Universidade de Toronto.

Resumo. *Treinamos uma rede neural convolucional grande para classificar as 1,2 milhões imagens de alta resolução no concurso ImageNet LSVRC-2010 para as 1000 diferentes classes. Nos dados de teste, atingimos as taxas de erro top-1 e top-5 de 37,5% e 17,0%, o que é consideravelmente melhor do que o estado da arte anterior. A rede neural, que tem 60 milhões de parâmetros e 650.000 neurônios, consiste em de cinco camadas convolucionais, algumas das quais são seguidas por camadas max-pooling, e três camadas totalmente conectadas com um softmax final de 1000 vias. Para fazer treinamento mais rápido, usamos neurônios não saturantes e uma implementação de GPU muito eficiente da operação de convolução. Para reduzir o overfitting nas camadas totalmente conectadas, empregamos um método de regularização recentemente desenvolvido chamado "dropout"Isso provou ser muito eficaz. Também inserimos uma variante desse modelo na Competição ILSVRC-2012 e alcançou uma taxa de erro de teste top-5 vencedora de 15,3%, comparado a 26,2% alcançado pela segunda melhor entrada.*

1. Introdução

As abordagens atuais para o reconhecimento de objetos fazem uso essencial dos métodos de aprendizado de máquina. Para melhorar seu desempenho, podemos coletar conjuntos de dados maiores, aprender modelos mais poderosos e usar técnicas melhores para evitar overfitting. Tarefas de reconhecimento simples podem ser resolvidas muito bem com conjuntos de dados desse tamanho, especialmente se forem aumentados com transformações de preservação de rótulo. Por exemplo, a taxa de erro atual mais alta na tarefa de reconhecimento de dígitos MNIST (¡0,3%) se aproxima do desempenho humano [4]. Mas objetos em ambientes realistas exibem variabilidade considerável, de modo que, para aprender a reconhecê-los, é necessário usar conjuntos de treinamento muito maiores. E, de fato, as deficiências dos conjuntos de dados de imagens pequenas têm sido amplamente reconhecidas (por exemplo, Pinto et al. [21]), mas recentemente só foi possível coletar conjuntos de dados rotulados com milhões de imagens. Os novos conjuntos de dados maiores incluem o LabelMe [23], que consiste em centenas de milhares de imagens totalmente segmentadas, e o ImageNet [6], que consiste em mais de 15 milhões de imagens de alta resolução rotuladas em mais de 22.000 categorias.

Para aprender sobre milhares de objetos de milhões de imagens, precisamos de um modelo com uma grande capacidade de aprendizado. No entanto, a imensa complexidade da tarefa de reconhecimento de objetos significa que esse problema não pode ser especificado nem mesmo por um conjunto de dados tão grande quanto o ImageNet, portanto nosso modelo também deve ter muitos conhecimentos prévios para compensar todos os dados que não temos. Redes neurais convolucionais (CNNs) constituem uma dessas classes de modelos [16, 11, 13, 18, 15, 22, 26]. Sua capacidade pode ser controlada variando

sua profundidade e amplitude, e eles também fazem pressuposições fortes e principalmente corretas sobre a natureza das imagens (ou seja, estacionariedade das estatísticas e localidade das dependências de pixel). Assim, em comparação com redes neurais feed-forward padrão com camadas de tamanho similar, as CNNs têm muito menos conexões e parâmetros e, portanto, são mais fáceis de treinar, enquanto seu desempenho teoricamente melhor provavelmente será apenas um pouco pior.

Apesar das qualidades atraentes das CNNs, e apesar da relativa eficiência de sua arquitetura local, elas ainda são proibitivamente caras de aplicar em larga escala a imagens de alta resolução.

As contribuições específicas deste trabalho são as seguintes: nós treinamos uma das maiores redes neurais convolucionais até hoje nos subconjuntos da ImageNet usados nas competições ILSVRC-2010 e ILSVRC-2012 [2] e alcançamos de longe os melhores resultados já relatados em esses conjuntos de dados. Escrevemos uma implementação de GPU altamente otimizada de convolução 2D e todas as outras operações inerentes ao treinamento de redes neurais convolucionais, que disponibilizamos publicamente1. Nossa rede contém vários recursos novos e incomuns que melhoram seu desempenho e reduzem seu tempo de treinamento, que são detalhados na Seção 3. O tamanho de nossa rede superou um problema significativo, mesmo com 1,2 milhão de exemplos de treinamento rotulados, então usamos vários técnicas efetivas para prevenir overfitting, que são descritas na Seção 4. Nossa rede final contém cinco camadas convolucionais e três totalmente conectadas, e essa profundidade parece ser importante: descobrimos que remover qualquer camada convolucional (cada uma delas contendo não mais que 1 % dos parâmetros do modelo) resultou em desempenho inferior.

2. O Dataset

O ImageNet é um conjunto de dados de mais de 15 milhões de imagens de alta resolução rotuladas, pertencentes a aproximadamente 22.000 categorias. As imagens foram coletadas da web e rotuladas por rotuladores humanos usando a ferramenta de crowdsourcing Mechanical Turk da Amazon. A partir de 2010, como parte do Pascal Visual Object Challenge, foi realizado um concurso anual chamado Desafio de Reconhecimento Visual de Grande Escala do ImageNet (ILSVRC). O ILSVRC usa um subconjunto do ImageNet com aproximadamente 1000 imagens em cada uma das 1000 categorias. Ao todo, existem aproximadamente 1,2 milhão de imagens de treinamento, 50.000 imagens de validação e 150.000 imagens de teste.

O ILSVRC-2010 é a única versão do ILSVRC para a qual os rótulos do conjunto de testes estão disponíveis, portanto, esta é a versão na qual realizamos a maioria das nossas experiências. Como também inserimos nosso modelo na competição ILSVRC-2012, na Seção 6 também relatamos nossos resultados sobre essa versão do conjunto de dados, para os quais os rótulos de conjunto de testes não estão disponíveis. No ImageNet, costuma-se relatar duas taxas de erro: top-1 e top-5, em que a taxa de erro top-5 é a fração de imagens de teste para as quais a etiqueta correta não está entre as cinco etiquetas consideradas mais prováveis pelo modelo .

ImageNet consiste em imagens de resolução variável, enquanto o nosso sistema requer uma dimensionalidade de entrada constante. Portanto, fizemos uma amostragem abaixo das imagens para uma resolução fixa de 256 256. Dada uma imagem retangular,

primeiro redimensionamos a imagem de modo que o lado mais curto fosse de 256, e então recortamos o patch central de 256 256 do resultado resultante. imagem. Nós não pré-processamos as imagens de qualquer outra forma, exceto pela subtração da atividade média sobre o conjunto de treinamento de cada pixel. Então nós treinamos nossa rede nos valores RGB brutos (centralizados) dos pixels

3. A arquitetura

Ela contém oito camadas aprendidas - cinco convolucionais e três totalmente conectadas. Abaixo, descrevemos alguns dos recursos novos ou incomuns da arquitetura da nossa rede. As Seções 3.1-3.4 são classificadas de acordo com a nossa estimativa de sua importância, com a mais importante primeiro.

3.1. Relu

A maneira padrão de modelar a saída f de um neurônio em função de sua entrada x é com

$$f(x) = tanh(x) ou f(x) = (1 + ex)1 \tag{1}$$

Em termos de tempo de treinamento com gradiente descendente, essas não-linearidades saturadas são muito mais lentas que a não-linearidade não-saturante

$$f(x) = max(0, x) \tag{2}$$

Seguindo Nair e Hinton [20], nos referimos aos neurônios com essa não-linearidade como Unidades Lineares Retificadas (ReLUs). Redes neurais convolucionais profundas com ReLUs treinam várias vezes mais rápido que seus equivalentes com unidades tanh. Mostra o número de iterações necessárias para atingir 25% de erro de treinamento no conjunto de dados CIFAR-10 para uma rede convolucional de quatro camadas específica. Este gráfico mostra que não teríamos sido capazes de experimentar com redes neurais tão grandes para este trabalho se tivéssemos usado modelos de neurônios de saturação tradicionais.

3.2. Overlapping Pooling

Camadas agrupadas em CNNs resumem as saídas de grupos vizinhos de neurônios no mesmo kernel mapa. Tradicionalmente, as vizinhanças resumidas por unidades de agrupamento adjacentes não se sobrepõem (por exemplo, [17, 11, 4]). Para ser mais preciso, uma camada de pooling pode ser considerada como uma grade de pooling unidades separadas por pixels, cada uma resumindo uma vizinhança de tamanho z z centrada no local da unidade de agrupamento. Se definirmos s = z, obtemos o agrupamento local tradicional como comumente empregado nas CNNs. Se definirmos s ¡z, obtemos pooling sobreposto. Isto é o que usamos em toda a nossa rede, com s = 2 ez = 3. Este esquema reduz as taxas de erro top-1 e top-5 em 0,4% e 0,3%, respectivamente, em comparação com o esquema não-sobreposto s = 2, z = 2, que produz saída de dimensões equivalentes. Em geral, observamos durante o treinamento que modelos com sobreposição de pool acham que é um pouco mais difícil de ajustar.

3.3. Arquitetura Geral

Agora estamos prontos para descrever a arquitetura geral da nossa CNN.Contém oito camadas com pesos; os cinco primeiros são convolucionais e os três restantes são totalmente conectado. A saída da última camada completamente conectada é alimentada para

um softmax de 1000 vias que produz uma distribuição sobre as etiquetas de 1.000 classes. Nossa rede maximiza o objetivo de regressão logística multinomial, que é equivalente a maximizar a média entre os casos de treinamento da probabilidade de log da etiqueta correta na distribuição de previsão. Os núcleos da segunda, quarta e quinta camadas convolucionais são conectados apenas àqueles mapas de kerner na camada anterior que residem na mesma GPU. Os núcleos da terceira camada convolucional são conectados a todos os mapas do kernel na segunda camada. Os neurônios nas camadas totalmente conectadas estão conectados a todos os neurônios da camada anterior. As camadas de resposta-normalização seguem as primeira e segunda camadas convolucionais. Camadas de pool máximo, do tipo descrito na Seção 3.4, seguem as duas camadas de normalização de resposta, assim como a quinta camada convolucional. A não-linearidade ReLU é aplicada à saída de cada camada convolucional e totalmente conectada.

neurônios em um mapa do kernel). A segunda camada convolucional toma como entrada a (resposta-normalizada e pooled) saida da primeira camada convolucional e filtra-a com 256 kernels de tamanho 5 5 48. A terceira, quarta e quinta camadas convolucionais são conectadas umas às outras sem nenhuma camada intermediária ou camadas de normalização. A terceira camada convolucional tem 384 núcleos de tamanho 3 x 3 x 256 ligados às saídas (normalizadas, reunidas) da segunda camada convolucional. A quarta camada convolucional tem 384 grãos de tamanho 3 3 192, e a quinta camada convolucional tem 256 grãos de tamanho 3 3 192. As camadas totalmente conectadas têm 4096 neurônios cada.

$$b^i_{x,y} = a^i_{x,y}\left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (\alpha_x, y^j)\right) \tag{3}$$

4. Reduzindo Overfiting

Nossa arquitetura de rede neural tem 60 milhões de parametros. Embora 1000 classes de ILSVRC façam com que cada exemplo de treinamento imponha 10 bits de restrição no mapeamento da imagem para o rótulo, isso tende a ser insuficiente para aprender tantos parametros sem considerar overfiting. Abaixo descrevemos as duas maneiras principais de combater o overfiting.

4.1. Aumento de Dados

O método mais comum e mais fácil para reduzir o overfiting em dados de imagem é aumentar artificialmente o conjunto de dados usando transformações de preservação de rótulos(por exemplo,[25,4,5]).Nós empregamos duas formas distintas de aumento de dados, ambos permitem que imagens sejam produzidas á partir da imagem original usando pouca computação, então as imagens transformadas não precisam ser armazendas no disco. Em nossa implementação, as imagens transformadas são geradas em código Python na CPU enquanto a GPU está treinando no lote anterior de imagens. Então esses esquemas de aumento de dados são, na verdade, livres computacionalmente.

A primeira forma de aumento de dados consiste em gerar traduções de imagens e reflexões horizontais. Fazemos isso extraindo aleatoriamente 224 224 patches (e suas reflexões horizontais)256 256 imagens e treinando nossa rede nesses patches extraídos4 .Isso aumenta o tamanho do nosso treinamento estabelecido por um fator de 2048, embora

os exemplos de treinamento resultantes sejam, é claro, altamente interdependentes. Sem este esquema, nossa rede sofre de overfitting substancial, o que nos obriga a usar redes muito menores. No tempo de teste, a rede faz uma previsão extraindo cinco patches de 224 224 (os quatro patches de canto e o patch central), bem como suas reflexões horizontais(dez patches ao todo), e calculando a média das previsões feitas pelo softmax da rede camada sobre os dez patches.

A segunda forma de aumento de dados consiste em alterar as intensidades dos canais RGB em imagens de treinamento. Especificamente, executamos o PCA no conjunto de valores de pixels RGB em todo o conjunto de treinamento do ImageNet. Para cada imagem de treinamento, adicionamos múltiplos dos componentes principais encontrados, com grandezas proporcionais aos autovalores correspondentes multiplicados por uma variável aleatória um gaussiano com média zero e desvio padrão de 0,1. Portanto, para cada pixel de imagem RGB

$$I_{x,y} = [I_{x,y}^{R}, I_{x,y}^{G}, I_{x,y}^{B}]^T \tag{4}$$

nós adicionamos a seguinte quantidade:

$$[P_1, P_2, P_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T \tag{5}$$

onde (Pi) e (i) são autovetor e autovalor da matriz de covariância 3 3 do pixel RGB valores, respectivamente, e (i) é a variável aleatória acima mencionada. Cada (i) é desenhado apenas uma vez para todos os pixels de uma determinada imagem de treinamento até que essa imagem seja usada para treinamento novamente, no ponto qual é re-desenhado. Este esquema captura aproximadamente uma propriedade importante de imagens naturais, ou seja, que a identidade do objeto é invariante para mudanças na intensidade e cor da iluminação. este esquema reduz a taxa de erro top-1 em mais de 1

4.2. Dropout

Combinando as previsões de muitos modelos diferentes é uma maneira bem-sucedida de reduzir erros de teste [1, 3], mas parece ser muito caro para grandes redes neurais que já levam vários dias treinar. Há, no entanto, uma versão muito eficiente da combinação de modelos que custa apenas cerca de fator de dois durante o treinamento. A técnica recém-introduzida, chamada “dropout” [10], consiste de definir a zero a saída de cada neurônio oculto com probabilidade 0,5. Os neurônios que são “dropped out” não contribuem para o passe para frente e não participam da retropropagação. Então toda vez que uma entrada é apresentada, a rede neural mostra uma arquitetura diferente, mas todas essas arquiteturas compartilham pesos. Esta técnica reduz co-adaptações complexas de neurônios, já que um neurônio não pode confiar na presença de outros neurônios em particular. Portanto, é forçado a aprender recursos mais robustos que são úteis em conjunto com muitos subconjuntos aleatórios diferentes do outros neurônios. No tempo de teste, usamos todos os neurônios, mas multiplicamos suas saídas por 0,5, o que é um aproximação razoável para tomar a média geométrica das distribuições preditivas produzidas pelas muitas redes de dropout exponencial.

Usamos dropout nas duas primeiras camadas totalmente conectadas. Sem dropout, nossa rede exibe overfitting substancial. O dropout praticamente dobra o número de iterações necessárias para convergir.

5. Reduzindo Overfiting

Nós treinamos nossos modelos usando gradiente descendente estocástico com um tamanho de lote de 128 exemplos, momento de 0,9 e peso decaimento de 0,0005. Descobrimos que esta pequena quantidade de decaimento de peso foi importante para o modelo aprender. Em outras palavras, o peso decadente aqui não é meramente um regularizador: reduz o erro de treinamento do modelo. A regra de atualização para peso (w) foi

$$\upsilon_i + 1 := 0.9 \cdot \upsilon_i - 0.0005 \cdot \epsilon \cdot \omega_i \cdot \epsilon \Big(\frac{\partial L}{\partial \omega} | \omega_i \Big) D_i \tag{6}$$

onde (i) é o índice de iteração, (v) é a variável de momento, é a taxa de aprendizado e

$$\omega_{i+1} := \omega_i + \upsilon_{i+1} \tag{7}$$

é a media superior do lote (Di) da derivada do objetivo em relação a (w), avaliado em (wi).

Inicializamos os pesos em cada camada a partir de uma distribuição gaussiana de média zero com desvio padrão 0,01. Inicializamos os vieses dos neurônios na segunda, quarta e quinta camadas convolucionais, bem como nas camadas ocultas totalmente conectadas, com a constante 1. Esta inicialização acelera os estágios iniciais de aprendizado, fornecendo às ReLUs entradas positivas. Nós inicializamos o neurônio vieses nas camadas restantes com a constante 0.

Usamos uma taxa de aprendizado igual para todas as camadas, que ajustamos manualmente durante o treinamento. A heurística que seguimos foi dividir a taxa de aprendizado por 10 quando o erro de validação da taxa parou de melhorar com a taxa de aprendizado atual. A taxa de aprendizagem foi inicializada em 0,01 e reduzida três vezes antes do término. Nós treinamos a rede por aproximadamente 90 ciclos através do conjunto de treinamento de 1,2 milhão de imagens, que levou de cinco a seis dias em duas GPUs NVIDIA GTX 580 de 3GB.

6. Resultados

Nossos resultados no ILSVRC-2010 estão resumidos na Tabela 1. Nossa rede alcança o top 1 e o top 5, taxas de erro do conjunto de testes de 37,5% e 17,0%. O melhor desempenho alcançado durante o ILSVRC2010 concorrência foi de 47,1% e 28,2% com uma abordagem que calcula a média das previsões de seis modelos de códigos esparsos treinados em diferentes características [2] e, desde então, os melhores os resultados são 45,7% e 25,7% com uma abordagem que calcula a média das previsões de dois classificadores treinado em Fisher Vectors (FVs) calculado a partir de dois tipos de características densamente amostradas [24]. Tabela 1: Comparação dos resultados no ILSVRC2010 Conjunto

Modelo	Top-1	Top-5
Sparse coding [2]	47.1%	28.2%
SIFT + FVs [24]	45.7%	25.7%
CNN	37.5%	17.0%

Tabela 1. Comparação de Resultados

de teste. Em itálico são melhores resultados alcançado por outros.

Também entramos no nosso modelo na competição ILSVRC-2012 e relatar nossos resultados na Tabela 2. Como os rótulos do conjunto de testes ILSVRC-2012 não estão publicamente disponíveis, não podemos reportar taxas de erro de teste para todos os modelos que nós tentamos. No restante deste parágrafo, usamos taxas de erro de validação e teste de forma intercambiável porque em nossa experiência eles não diferem em mais de 0,1% (veja a Tabela 2). A CNN descrita neste documento alcança uma taxa de erro de 5% de 18,2%. Média das previsões de cinco CNNs similares dá uma taxa de erro de 16,4%.

Treinando uma CNN, com um sexto extra convolucional camada sobre a última camada de pooling, para classificar todo o lançamento do ImageNet Fall 2011 (15M imagens, 22K categorias), e depois "afinar" no ILSVRC-2012 resulta uma taxa de erro de 16,6%. Média das previsões de duas CNNs que foram pré-treinadas em todo o lançamento do outono de 2011 com as supracitadas cinco CNNs, apresenta uma taxa de erro de 15,3%. O segundo melhor alcançou uma taxa de erro de 26,2% com uma abordagem que calcula a média das previsões de vários classificadores treinados em FVs computados a partir de diferentes tipos de características densamente amostradas.

[TABELA]Tabela 2: Comparação das taxas de erro na validação do ILSVRC-2012 conjuntos de teste. Em itálico, os melhores resultados são alcançados por outros. Modelos com um asterisco * foram "pré-treinados" para classificar toda a ImageNet 2011 Fall lançamento. Veja a seção 6 para detalhes.

Finalmente, também relatamos nosso erro taxas sobre a versão do outono de 2009 ImageNet com 10.184 categorias e 8,9 milhões de imagens. Nisto conjunto de dados seguimos a convenção na literatura de usar metade do as imagens para treinamento e meio para testes. Como não há conjunto de testes, nossa divisão necessariamente difere das divisões usadas por autores anteriores, mas isso não afeta os resultados apreciavelmente. Nossas taxas de erro top-1 e top-5 neste conjunto de dados são 67,4% e 40,9 % , obtidos pela rede descrita acima, mas com uma sexta camada convolucional adicional sobre a última camada de pooling. Os melhores resultados publicados neste conjunto de dados são 78,1% e 60,9%[19].

6.1. Avaliação Qualitativa

A rede aprendeu uma variedade de núcleos seletivos de freqüência e orientação, bem como vários blobs. Observe a especialização exibida pelas duas GPUs, resultado da conectividade restrita descrito na Seção 3.5. Os núcleos no GPU 1 são em grande parte agnósticos, enquanto os núcleos na GPU 2 são em grande parte específicos de cor. Este tipo de especialização ocorre durante cada corrida e é independente de qualquer inicialização específica de peso aleatório (módulo de renumeração das GPUs).

Avaliamos qualitativamente o que a rede aprendeu computando sua previsões top-5 em oito imagens de teste. Observe que mesmo objetos fora do centro, como o ácaro no canto superior esquerdo, pode ser reconhecido pela rede. A maioria dos top-5 rótulos parece razoável. Por exemplo, apenas outros tipos de gatos são considerados rótulos plausíveis para o leopardo. Em alguns casos (grade, cereja) existe uma genuína ambiguidade quanto ao foco pretendido da fotografia. Outra maneira de investigar o conhecimento visual da rede é considerar as ativações de recurso induzidas por uma imagem na última camada oculta de 4096 dimensões. Se duas imagens produzem vetores de ativação de feições com uma pequena separação euclidiana, podemos dizer que os níveis mais altos

da rede neural os consideram semelhantes. Observe que, no nível de pixel, a tensão sofrida em relação a uma imagem geral geralmente não está presente em L2 para consultar as imagens na primeira coluna. Por exemplo, os cães e elefantes recuperados aparecem em uma variedade de poses. Apresentamos os resultados para muitas outras imagens de teste no material suplementar. A similaridade computacional usando a distância Euclideana entre dois vectores de 4096 dimensões e valor real é suficiente, mas pode ser eficiente se a transmissão automática apressar os vetores para codificar os códigos binários. Eles devem produzir um método mais adequado para o imageamento do que aplicar os codificadores aos pixels brutos[14], que não faz uso de rótulos de imagem e, portanto, tem a tendência de recuperar imagens com padrões semelhantes de bordas, sejam semanticamente semelhantes.

7. Conclusão

Nossos resultados mostram que uma rede neural convolucional grande e profunda é capaz de alcançar resultados recordes em um conjunto de dados altamente desafiador, usando aprendizado puramente supervisionado. É notável que o desempenho da nossa rede se degrada se uma única camada convolucional for removida. Por exemplo, remover qualquer uma das camadas intermediárias resulta em uma perda de cerca de 2% para o desempenho da primeira rede. Portanto, a profundidade é realmente importante para alcançar nossos resultados. Para simplificar nossos experimentos, não usamos nenhum pré-treinamento não supervisionado, embora esperemos que ajude, especialmente se obtivermos poder computacional suficiente para aumentar significativamente o tamanho da rede sem obter um aumento correspondente na quantidade de dados rotulados. Até agora, nossos resultados melhoraram à medida que aumentamos nossa rede e a treinamos por mais tempo, mas ainda temos muitas ordens de grandeza para acompanhar o caminho infero-temporal do sistema visual humano. Em última análise, gostaríamos de usar redes convolucionais muito grandes e profundas em seqüências de vídeo em que a estrutura temporal fornece informações muito úteis que estão faltando ou muito menos óbvias em imagens estáticas.

Referências

[1] R.M.BellandY.Koren.Lessonsfromthenetixprizechallenge. ACMSIGKDDExplorationsNewsletter, 9(2):75–79, 2007.

[2] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.imagenet.org/challenges. 2010.

[3] L. Breiman. Random forests. Machine learning, 45(1):5–32, 2001. [4] D. Cires¸an, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classication. Arxiv preprint arXiv:1202.2745, 2012.

[5] D.C. Cires¸an, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classication. Arxiv preprint arXiv:1102.0183, 2011.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[7] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012, 2012. URL http://www.image-net.org/challenges/LSVRC/2012/.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesianapproachtestedon101objectcategories. Computer Vision and Image Understanding, 106(1):59–70, 2007.

[9] G. Grifn, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. URL http://authors.library.caltech.edu/7694.

[10] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[11] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In International Conference on Computer Vision, pages 2146–2153. IEEE, 2009.

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.

[13] A. Krizhevsky. Convolutional deep belief networks on cifar-10. Unpublished manuscript, 2010.

[14] A. Krizhevsky and G.E. Hinton. Using very deep autoencoders for content-based image retrieval. In ESANN, 2011.

[15] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Handwrittendigitrecognitionwithaback-propagationnetwork. InAdvancesinneuralinformationprocessing systems, 1990.

[16] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 2, pages II–97. IEEE, 2004.

[17] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.

[18] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervisedlearningofhierarchicalrepresentations. InProceedingsofthe26thAnnualInternationalConference on Machine Learning, pages 609–616. ACM, 2009.

[19] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric Learning for Large Scale Image Classication: Generalizing to New Classes at Near-Zero Cost. In ECCV - European Conference on Computer Vision, Florence, Italy, October 2012.

[20] V. Nair and G. E. Hinton. Rectied linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning, 2010.

[21] N. Pinto, D.D. Cox, and J.J. DiCarlo. Why is real-world visual object recognition hard? PLoS computational biology, 4(1):e27, 2008.

[22] N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation.

PLoS computational biology, 5(11):e1000579, 2009.

[23] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. International journal of computer vision, 77(1):157–173, 2008.

[24] J.SánchezandF.Perronnin. High-dimensionalsignaturecompressionforlarge-scaleimageclassication. InComputerVisionandPatternRecognition(CVPR),2011IEEEConferenceon,pages1665–1672.IEEE, 2011.

[25] P.Y. Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional neural networks applied to visualdocumentanalysis. InProceedingsoftheSeventhInternationalConferenceonDocumentAnalysis and Recognition, volume 2, pages 958–962, 2003.

[26] S.C.Turaga,J.F.Murray,V.Jain,F.Roth,M.Helmstaedter,K.Briggman,W.Denk, andH.S.Seung. Convolutional networks can learn to generate afnity graphs for image segmentation. Neural Computation, 22(2):511–538, 2010.