UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Diogo Felipe Félix de Melo

Aprendizado profundo com capacidade computacional reduzida: uma aplicação à quebra de captchas.

Recife

2015

Diogo Felipe Félix de Melo

Aprendizado profundo com capacidade computacional reduzida: uma aplicação à quebra de captchas.

Monografia apresentada ao Curso de Bacharelado em Ciências da Camputação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciências da Camputação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Bacharelado em Ciências da Camputação

Orientador: Pablo de Azevedo Sampaio

Recife

2015

Agradecimentos

Meus pais, familiares e amigos.

Ao meu orientador, por toda a paciência e dedicação.

Resumo

Palavras-chave: Aprendizado de Maquina, Aprendizado Profundo, CAPTCHA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CAPTCHA	Completely Automated Public Turing tests to tell Computers and Humans ApartUser Datagram Protocol.
OCR	Optical Character Recognition.
RGB	Red-Green-Blue. Canais de codificação de cores.
RAM	Random Access Memory. Memória de Acesso Aleatório.
HIP	Human Interaction Proofs. Provas de interação humana.

Sumário

1 Introdução

Modelos de aprendizado baseados em neurologia são conhecidos desde meados do século passado(1). Das proposições iniciais até os dias de hoje, essa classe modelos tem evoluído em complexidade e técnicas de forma contínua, culminando em modelos com muitas camadas e níveis cada vez mais abstratos de representações (ver (2) para uma breve revisão histórica). Os poucos resultados teóricos disponíveis demonstram que essa classe de modelos possuem um alto poder de expressividade, sendo capaz de, sob certas circunstâncias, representar diversas classes de funções(3, 4). Apesar dos avanços na área, foi apenas recentemente que modelos neurais começaram a redefinir o estado da arte, superando outras classes de algoritmos de aprendizado de máquina(5) e até mesmo alcançando performances sobre humanas(6). Tais avanços foram possíveis devido a três fatores chaves: a viabilização de bases de treino cada vez maiores, o aumento do poder computacional e o desenvolvimento de novas arquiteturas e técnicas de treino.

A crescente melhoria de performance dos modelos de aprendizado profundo tem motivado estudos em áreas onde se é preciso distinguir computadores e humanos. CAPTCHAs (7) (do inglês Completely Automated Public Turing tests to tell Computers and Humans Apart) definem uma coleção de técnicas que tem como objetivo bloquear a ação de agentes autônomos na rede mundial de computadores. O subconjunto mais conhecido dessas técnicas talvez seja o de CAPTCHAs baseados em texto(8). Nesse tipo de desafio, uma imagem contendo uma sequência de caracteres é exibida. A validação é feita pela comparação entre o texto informado pelo usuário e a resposta correta. Em trabalhos recentes, foram relatadas acurácias próximos à humana em sequências formadas exclusivamente por números(9) ou por uma única fonte(10). Para o problema geral de quebrar captchas baseados em texto, entretanto, modelos de aprendizado profundo ainda mostram desempenho inferior ao humano. Contudo, pesquisas recentes apontam para avanços claros nos próximos anos(11). Em comum, esses modelos possuem a necessidade de muito poder computacional e/ou bases de dados extensivas. O treino dessas redes é tipicamente executado em clusters e/ou sistemas de computação sob demanda, com alto poder de paralelização e utilizando hardware de alto poder de processamento como GPUs e TPUs. Adicionalmente, as bases de treino comumente alcançam alguns terrabytes e envolvem grandes operações de aquisição e/ou geração.

Neste trabalho propomos uma abordagem comparativa entre diferentes arquiteturas de redes neurais para a solução de CAPTCHAs baseados em texto, nos restringindo, entretanto, à um ambiente com poder computacional reduzido. Pretendemos mostrar que é possível fazer uso dessas técnicas em computadores pessoais e ainda obter resultados próximos ao estado da arte encontrado na literatura. Este trabalho se encontra organizado

como segue. No capítulo 2 apresentamos uma breve introdução sobre diferentes tipos de CAPTCHAs, com ênfase em desafios baseados em texto. Sequencialmente, arquiteturas e técnicas de redes neurais são apresentadas. No capítulo 3, O problema de extração de CAPTCHAs baseados em texto é formulado matematicamente e os principais resultados da literatura na solução do problema são comparados. Por fim, uma descrição da arquitetura dos modelos usados neste estudo é feita. No Capítulo 4, detalhes dos experimentos realizados são formalizados. No capítulo 5, os resultados dos experimentos são comparados e nossas conclusões apresentadas.

2 Fundamentação

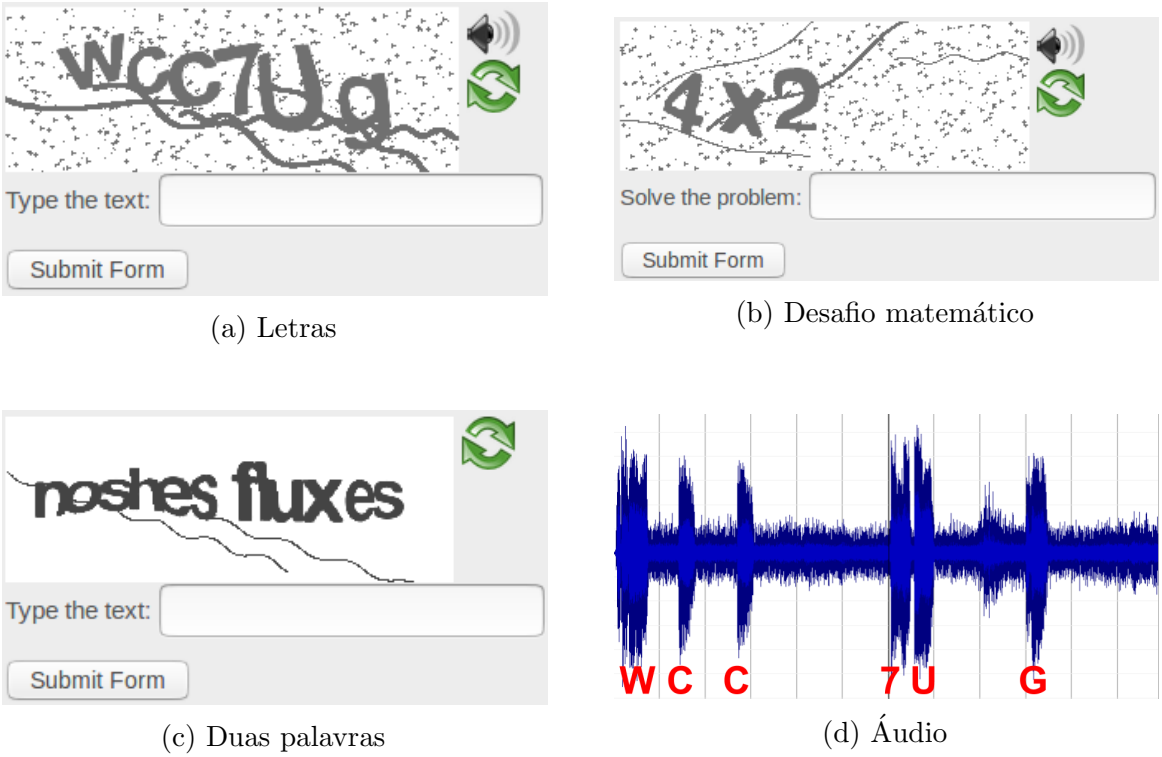
Neste capítulo abordaremos como funcionam os principais tipos de CAPTCHA conhecidos, dando um enfoque especial aos captchas baseados em texto. Em seguida, redes neurais são introduzidas como um problema de minimização e as técnicas de treino utilizadas neste trabalho são descritas.

2.1 CAPTCHAS

CAPTCHAs(7) ou HIP (do inglês Human Interaction Proofs), são um conjunto de técnicas que tem como objetivo discernir a ação automatizada de robôs da ação de serem humanos na Rede Mundial de Computadores. Esses filtros tem sido usados de forma efetiva em diversas aplicações: proteger informações sensíveis, como *e-mail* e dados pessoais; impedir tentativas de *login* automatizados; acesso massivo a sistemas de bases de dados entre outros. Entretanto, desde as primeiras aplicações até os dias de hoje, existe uma corrida co-evolucionária entre atacantes e defensores. Por um lado, algoritmos de '*quebra*' de CAPTCHA se tornam cada vez mais sofisticados e precisos. Por outro lado, filtros mais complexos são desenvolvidos. Entretanto, como explicado por (12), existe um balanço entre complexidade e factibilidade que os defensores devem buscar, explorando habilidades em que humanos ainda não foram ultrapassados por máquinas.

De forma geral, esses filtros podem ser formulados como um desafio sobre um conjunto de domínio cuja a resposta é um token. O domínio pode ser um trecho de áudio, uma sequencia de imagens ou até mesmo o histórico de navegação do desafiado. O token pode ser constituído de um conjunto de ações, o texto extraído de um áudio ou imagem, ou possuir um histórico de navegação de baixo risco. Podem ainda ser constituídos de uma única etapa ou de varias. Na figura 1 podemos ver diferentes tipos de CAPTCHAs gerados com a biblioteca de código aberto *Securimage* (13).

Em CAPTCHAs baseados em texto, uma imagem contendo uma sequência de caracteres é exibida ao desafiado. O teste consiste em conseguir recuperar corretamente o texto. Quanto ao texto, diferentes complexidades podem ser adicionadas: variar em tamanho, de alguns poucos a vários caracteres; ser composto por mais de uma palavra; utilizar diferentes alfabetos, apenas números, alfa-numérico, símbolos, diferenciar maiúsculas de minúsculas etc.; utilizar diferentes fontes, inclusive objetos que se assemelham a letras ou formas geométricas. Os caracteres podem ser sorteados aleatoriamente a partir de um conjunto sem correlação entre sí ou serem oriundos de um repositório de palavras. Neste último caso, o desafio se torna mais fácil para humanos, dada nossa característica em reconhecer padrões, mas pode representar uma fraqueza caso a base seja exposta. Quanto



(a) Letras

(b) Desafio matemático

(c) Duas palavras

(d) Áudio

Figura 1 – Diferentes tipos de CAPTCHAs. (a) O desafiante deve reconhecer corretamente um texto formado por caracteres não correlacionados. (b) O desafiado deve resolver um problema simples de álgebra. (c) Palavras sorteadas de um repositório. (d) Reconhecimento de áudio. Gerados com a biblioteca Securimage.

a imagem, usualmente se utiliza adição de ruído, linhas e/ou grades para dificultar o processo de segmentação de caracteres (12), efeitos de distorção como corrosão ou dilatação, transformações geométricas como rotação e translação, entre outros.

Um exemplo da co-evolução desses sistemas é a forma como o reCaptcha (14) tem mudado ao longo dos anos. Quando proposto, o sistema era baseado em trechos de livros e jornais antigos que haviam falhado em processos de OCR (do inglês optical character recognition). Trechos que não haviam sido corretamente identificados (teste) eram separados e exibidos para humanos juntamente com imagens cuja a extração era reconhecida (controle). Teste e controle eram comparados para certificar a atuação humana. Quando proposto, os autores alegaram que humanos eram capazes de resolver o desafio em quase todos os casos e que comutadores teriam chance quase nula de passarem desapercebidos, já que o repositório de exemplos era composta por imagens em que os melhores algoritmos haviam falhado. Porém, poucos anos depois, avanços em OCR obtiveram 99% de precisão na base de texto utilizada por esse sistema (9), inviabilizando o uso dessa técnica e forçando o reCaptcha a evoluir para uma segunda versão. Nessa nova abordagem, os dados de navegação dos usuários são analisados por um algoritmo de risco. Caso uma pontuação mínima seja atingida, o usuário é considerado humano, caso contrário, é exposto a problemas conhecidamente difíceis para computadores, como reconhecimento de objetos,

contextualização de imagens e busca de similaridades, combinados com diferentes ações que devem ser performadas pelo usuário. Entretanto, mesmo essa nova versão pôde ser enganada por robôs em 85% das vezes (15), forçando a constante atualização dos desafios.

Em geral, o processo de '*quebra*' envolve duas ideias principais: explorar vulnerabilidades ou uso de algoritmos inteligentes. Por serem testes automatizados, CAPTCHAs geralmente apresentam algum padrão de comportamento ou falha de projeto. A padronização na geração de desafios pode permitir que um atacante desenvolva heurísticas de ataque. Imagens com mesmo espaçamento de caracteres ou padrões repetitivos podem ser explorados e facilitar a segmentação da imagem, como foi explorado por (16), por exemplo. Falhas ou vieses no projeto de algoritmos também podem expor brechas. Nas primeiras versões do reCaptcha, por exemplo, possuir dados de navegação antigos era um dos principais critérios de avaliação de risco. De fato, essa foi a brecha explorada por (15) para conseguir uma alta taxa de sucesso. Quanto ao uso de algoritmos inteligentes, redes neurais recebem um lugar de destaque devido a flexibilidade e capacidade de generalização que esses modelos conseguem alcançar. O uso dessa classe de algoritmos foi a abordagem escolhida por (9) e (12), por exemplo. Redes neurais são exploradas na sequência.

2.2 Redes Neurais

Dentro do campo de Inteligência Artificial, aprendizado de máquina é uma técnica que permite aprender as regras de um problema de forma autônoma, baseado-se apenas em exemplos. Uma categoria especial é a de aprendizado supervisionado, onde os exemplos constituem-se de um *elemento* em um domínio conhecido e um *rótulos* associado. O objetivo é aprender regras que permitam relacionar elementos e rótulos. Uma rede neural[1] é, em última análise, uma forma genérica de escrever funções[2] sobre relações elementos-rótulos, transformando o problema de encontrar regras em um problema de minimização numérica. O adjetivo '*neural*' adivem da inspiração em funções biológicas que historicamente inspiraram e ainda inspiram essas funções.

De forma mais específica, dado um conjunto de exemplos $D = \{(x, y)\}$, onde x pertence ao domínio conhecido e y um rótulo associado, desejamos encontrar a função $\hat{y} = f(x)$, de tal modo que $\hat{y}$ seja o mais similar possível à y. Por '*mais similar o possível*' entende-se que conhecemos uma função de erro, também referida como função custo, que é tão menor quanto melhor for a aproximação dada por $f(x)$, e é normalmente representada como $J(y, \hat{y})$. Formalmente, desejamos encontrar f^* tal que

$$f^* = \min_f J^{(D)} = \langle J(y, \hat{y}) \rangle_D, \tag{2.1}$$

[1] Apesar de introduzidas aqui como um algoritmo supervisionado, redes neurais podem aplicadas à problemas não supervisionados.

[2] Funções estão sendo usadas aqui com um sentido mais relaxado do que o usualmente utilizado na matemática.

onde $\langle \ldots \rangle_D$ representa o valor esperado no conjunto D.

Redes neurais são um conjunto de técnicas inspiradas em processos cognitivos desempenhados pelo sistema nervoso que fornecem uma maneira de descrever famílias de funções. Dada uma família de funções $f^{\Theta} : x \to y$ definida por uma rede neural e parametrizadas por Θ, podemos vasculhar o espaço de busca induzido por $\{\Theta\}$ para encontrar um função que satisfaça alguma propriedade de interesse. Em particular, no caso de aprendizado de máquina, estamos interessados em encontrar o parâmetro Θ^* tal que:

$$\Theta^* = \min_{\Theta} \langle J(y, f^{\Theta}(x)) \rangle_D. \tag{2.2}$$

Podemos utilizar composição de funções para construir redes neurais mais complexas e expressivas:

$$f^{\Theta}(x) = f^{\Theta_1}(f^{\Theta_2}(f^{\Theta_3}(\ldots(x)))), \tag{2.3}$$

sendo $\Theta = (\Theta_1, \Theta_2, \Theta_3, \ldots)$. Quando compomos funções desta forma, é comum nos referirmos à cada função f^{Θ_i} como sendo a *i-ésima camada* da rede neural, sendo as camadas para além da mais externa também conhecidas como *camadas escondias*. A *profundidade* da rede é uma referência à quantidade de funções internas usadas na composição. Diferentes tipos de funções definem diferentes tipos de transformações, as quais nos referimos como tipo da camada. A especificação de todas as camadas em uma rede neural é o que chamamos de *arquitetura* da rede. A seguir vamos explorar dois tipos de camadas que foram utilizados no presente estudo.

2.2.1 Camadas Densas

Camadas *densas* ou totalmente conectadas definem uma transformação afim entre o conjunto de entradas e saídas. Tipicamente, após a transformação afim, segue-se a aplicação de uma função não linear elemento-à-elemento, conhecida como *função de ativação*, permitindo a expressão de relações mais complexas entre esses dois conjuntos. As camadas densas são biologicamente inspiradas no mecanismo de comunicação dos neurônios, onde a diferença de potencial elétrico experimentado nos axônios é proporcional, em alguma medida, à soma das diferenças de potenciais nos dendritos.

De maneira mais formal, seja x um vetor no conjunto de entrada, a relação expressa por uma camada densa é dada por:

$$f^{W,b}(x) = act(W \odot x + b) \tag{2.4}$$

onde b é um vetor, referido como *viés*, W uma matriz de transformação, act uma função de ativação e $\odot$ é a operação usual de multiplicação de matrizes, definida elemento-à-elemento como $[W \odot x]_i = \sum_k W_{ik} * x_k$. Diferentes funções de ativação expressam diferentes não linearidades. Dentre os exemplos mais conhecidos na literatura, ressaltamos a função

sigmoide $\sigma(z)_i = \frac{1}{1+\exp(-z_i)}$, que mapeia os elementos de saída no intervalo $\Re[0,1]$, a função de retificação linear (relu), definida por $relu(z)_i = max(0, z_i)$, e a função softmax, $\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$, que possui a interessante propriedade $\sum_i \sigma(z)_i = 1$, sendo usualmente utilizada para expressar distribuições de probabilidade.

2.2.2 Camadas Convolucionais

Camadas *convolucionais*

$$f^{W,b}(x) = act(W \otimes x + b) \tag{2.5}$$

$$[W \otimes x]_{ijd} = \sum_{m=i'-k_i, n=j'-k_j}^{i'+k_i, j'+k_j} W_{m,n,c,d} * x_{m,n,c} \tag{2.6}$$

onde k_i e k_j são o tamanho do núcleo da transformação e a relação entre $s_i = i'/i$ ($s_j = j'/j$) definem o passo da transformação (em geral, $k_i = k_j = k$ e $s_i = s_j = s$). W é um tensor de ranque 4 e b um *offset* com dimensão igual à última dimensão de W.

O núcleo da transformação define um campo perceptivo

2.3 Aprendizado e Normalização

Gradiente Descente, Rprop, Adam

Dropout, L2, L1, Batch-Norm

3 Modelagem

Nesta secção iremos definir de forma mais precisa o problema de extração de tokens dos CAPTCHAS de texto e sua formulação como um problema de minimização. Os resultados recentes encontrados na literatura serão abordados e comparados com a formulação proposta nesse trabalho. No final da secção as arquiteturas de rede utilizadas nos experimentos serão introduzidas.

Definição do problema

Matematicamente, uma imagem com altura H, largura W e C canais pode ser representada como um tensor $x \in \Re^{HxWxC}$, onde H, C e C são, respectivamente, a altura o comprimento e o número de canais da imagem. O token é uma sequencia w sob um alfabeto Σ. O desafiante passa na tarefa de acertar cada elemento w_i da sequencia.

estado da arte

CAPTCHAs de texto(8) podem ser vistos como um problema de extração de texto em imagens, sendo assim uma generalização para o problema de OCR. É preciso ressaltar, entretanto, que essas imagens são especialmente desenvolvidas para serem de difícil solução para computadores e preferencialmente fáceis para seres humanos. Assim, algoritmos usuais de OCR tendem a demonstrar baixo desempenho na solução desses desafios.

antigo quebrar captcha (12) resultados 50 porcento uso massivo de reconhecimento de caracteres

definição das redes definição da nomenclatura

4 Metodologia

Neste capítulo os detalhes envolvidos na geração das imagens de CAPTCHAs são expostos. Em seguida, definimos as grandezas de interesse que nos permitem acessar a qualidade dos modelos treinados. Por fim, as etapas de treino e validação são formalizadas.

4.1 Geração dos CAPTCHAs

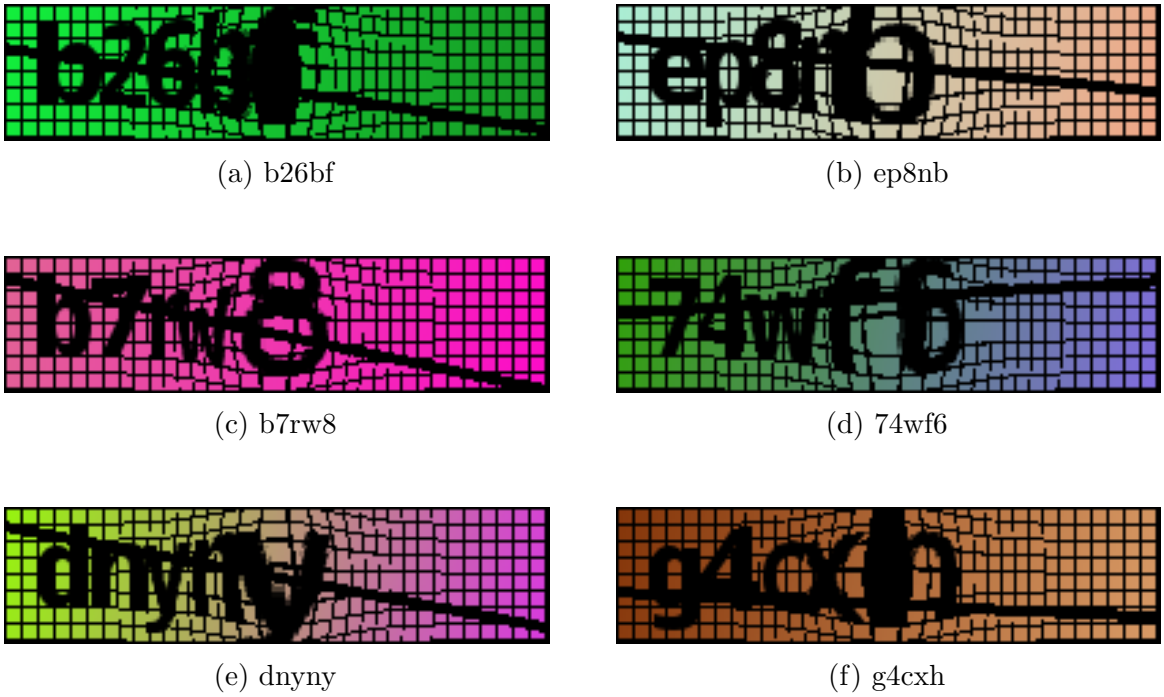
Todos os exemplos foram gerados utilizando a biblioteca SimpleCaptcha(17). Ao total, foram gerados 30000 pares imagem-token, diferentes cores e efeitos e tokens sob o alfabeto (ordenado) $\Sigma = \{0123456789abcdefghijklmnopqrstuvwxyz\}$ com comprimento fixo em 5. Dentre os efeitos escolhidos, enfatizamos que variações nas cores de fundo, desenho de grades, adição de linhas aleatórias e deformação em explosão são técnicas efetivas para construir desafios fáceis para humanos e difíceis para computadores de acordo com estudo conduzido por (12). Uma pequena amostra das imagen-token geradas pode ser vista na Fig.2.

Considere $D = (x, y)$ o conjunto formado por todos os pares de imagem-token gerados. Cada exemplo é formado por um tensor imagem x e uma matriz y representando o token, de dimensões $(200, 50, 3)$ e $(5, 36)$, respectivamente. Cada entrada $x_{ijk} \in \Re[0, 1]$ representa a intensidade do pixel localizado na posição (i, j) e canal k. A entrada $y_{ij} \in \Re[0, 1]$ foi codificada utilizando-se a técnica *one-hot encoding*, onde i representa a posição na sequência w e j o índice no vocabulário do caractere nessa posição, de modo que

$$y_{ij} = \begin{cases} 1, & \text{se } w_i = \Sigma_j \\ 0, & \text{caso contrário.} \end{cases} \tag{4.1}$$

Essa codificação nos permite interpretar y como sendo uma distribuição de probabilidade. Seja $ord(w_i)$ o índice no vocabulário tal que $w_i = \Sigma_{ord(w_i)}$ e c o caractere em w_i, temos que $y_{ij=ord(c)} = p(w_i = c|x) = 1$ e $y_{i\, j\neq ord(c)} = p(w_i \neq c|x) = 0$, isto é, temos 100% de certeza de que o caractere em w_i é c.

D foi reordenando de forma aleatória e separado em dois subconjuntos: o conjunto de treino, D_{tr}, com $\frac{2}{3}$ do total de pares, e o conjunto de validação, D_{val}, com os demais exemplos. Devido a natureza combinatória do espaço de imagens possíveis (36^5 tokens $\times$ 255^3 cores de fundo $\times$ espaço de todas as pertubações possíveis), acreditamos que não existam exemplos em comum nesses dois conjuntos.



(a) b26bf (b) ep8nb

(c) b7rw8 (d) 74wf6

(e) dnyny (f) g4cxh

Figura 2 – Exemplos de CAPTCHAs gerados e seus respectivos tokens.

4.2 Treino e Validação

Em todos os experimentos as redes foram inicializadas segundo a heurística proposta em (18). Após a inicialização, seguem-se repetidas épocas de aprendizado até que o critério de parada (descrito mais adiante) seja satisfeito ou um máximo de T^{max} épocas seja atingido.

Uma época de aprendizado consiste em duas etapas: treino e validação. Durante o treino, um subconjunto $D_{batch} \subset D_{tr}$ é sorteado ao acaso. Os parâmetros da rede são atualizados utilizando o algoritmo Adam(19) com taxa de aprendizado l_r de forma a minimizar o erro nesse subconjunto. A etapa de treino se encerra após $|D_{tr}|/|D_{batch}|$ atualizações. Na etapa de validação, as grandezas de interesse são calculadas para D_{tr} e D_{val} e salvas para posterior análise.

Para selecionar o valor do hiper-parâmetro l_r, foram realizados experimentos usando diferentes valores de l_r e $T^{max} = 10$ épocas para cada arquitetura. A partir dos experimentos, selecionamos manualmente os limites inferior e superior, (l_r^-, l_r^+), que apresentam o melhor compromisso entre velocidade de aprendizado e estabilidade. O experimento é então executado novamente utilizando decaimento linear para l_r de acordo com a equação:

$$l_r(t) = l_r^+ + (l_r^- - l_r^+) * \frac{t}{T_{max} - 1}, \tag{4.2}$$

onde $t = 0, 1, 2, \ldots, T_{max} - 1$, onde t é a época atual.

O critério de parada é definido por uma heurística semelhante às definidas por (20). O aprendizado leva no mínimo 10 e no máximo 50 épocas. Após a etapa de validação o aprendizado é interrompido prematuramente se um dos dois critérios forem verificados: o erro calculado em D_{val} na época atual ultrapasse em mais de 10% o menor valor do erro nesse conjunto nas épocas anteriores; o valor do erro atual em D_{tr} seja maior do que 97% da média dos cinco últimos erros nesse conjunto. Ou seja, o treinamento é parado prematuramente se for detectado *overfit* ou se não houver melhora significativa em relação aos últimos valores.

Todos os experimentos realizados nesse trabalho foram executados em um pentium core i5 com 8gb de RAM utilizando a biblioteca tensorflow. Sendo este o limite prático para o tamanho dos modelos e da base de treino.

4.3 Grandezas de interesse

Na secção de fundamentação teórica de redes neurais (sec. 2.2), vimos que cada arquitetura é parametrizada Θ. Mais especificamente, cada uma das arquiteturas utilizadas neste trabalho possui como parâmetros um conjuntos de números reais. Assim, definimos a complexidade do modelo, para fins de comparação, como a soma da quantidade de parâmetros de cada camada da arquitetura. Para treinar e acessar a qualidade dos modelos, consideramos as grandezas definidas à seguir. Para todas as definições, considere D um conjunto de exemplos, $(x, y) \in D$ e $\hat{y} = f^{\Theta}(x)$ a distribuição de probabilidade inferida, como descrito anteriormente.

A entropia cruzada pode ser interpretada como uma medida de divergência entre duas distribuições de probabilidade. O erro associado ao inferir $\hat{y}$ quando a verdadeira distribuição deveria ser y, por caractere, é dado por

$$H_i(y, \hat{y}) = -\sum_j y_{ij} \log_2 \hat{y}_{ij} \tag{4.3}$$

$$= -\log_2 \hat{y}_{i\, ord(wi)} \tag{4.4}$$

onde utilizamos o fato de $y_{ij} = 0$ exceto em $j = ord(w_i)$. Em outras palavras, a entropia associada ao classificador da posição i é o logaritmo da probabilidade predita para o caractere correto nessa posição. Definimos o erro esperado do classificador i no subconjunto D

$$J_i^{(D)} = \frac{1}{|D|} \sum_{(x,y)\in D} H_i(y, \hat{y}). \tag{4.5}$$

e erro total de predição do token com a soma dos erros em cada posição:

$$J^{(D)} = \sum_i J_i^{(D)} \tag{4.6}$$

Durante o treino tentaremos minimizar a 4.5 para cada classificador, e o erro total do modelo é estimado pela equação 4.6.

Uma estimativa da probabilidade de acerto por caractere é dada pela acurácia de cada classificador, isto é, o número de acertos do caractere i no conjunto D, N_i, normalizado pelo tamanho do conjunto D:

$$\hat{p}_i^{(D)} = acc_i^{(D)} = \frac{N_i}{|D|} \tag{4.7}$$

Supondo que os $\hat{p}_i^{(D)}$ sejam independentes entre si, podemos definir uma estimativa para a probabilidade de acerto do token como o produto das probabilidades individuais:

$$\hat{p_w}^{(D)} = \prod_i \hat{p}_i^{(D)}. \tag{4.8}$$

Adicionalmente, definimos a acurácia do modelo como sendo o número de acertos na predição do token, N_w, normalizado pelo tamanho do conjunto:

$$acc_w^{(D)} = \frac{N_w}{|D|}. \tag{4.9}$$

Chamamos a atenção de que as equações 4.8 e 4.9 não necessariamente representam a mesma grandeza, fornecendo duas estimativas diferentes para a qualidade do modelo.

Quanto ao tempo dos algoritmos, estamos interessados em duas medidas: o tempo de treino por época, $\tilde{\tau}$, o tempo total (treino e validação) por época, τ, e o tempo até convergência T.

5 Resultados

Referências

1 ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958. Citado na página 8.

2 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado na página 8.

3 BARRON, A. R. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Information Theory*, v. 39, p. 930–945, 1993. Citado na página 8.

4 ANDONI, A. et al. Learning polynomials with neural networks. v. 5, p. 3955–3963, 01 2014. Citado na página 8.

5 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: ⟨http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf⟩. Citado na página 8.

6 MNIH, V. et al. Human-level control through deep reinforcement learning. *Nature*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529–533, fev. 2015. ISSN 00280836. Disponível em: ⟨http://dx.doi.org/10.1038/nature14236⟩. Citado na página 8.

7 AHN, L. von et al. Captcha: using hard ai problems for security. In: *Advances in Cryptology, Eurocrypt*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. v. 2656, p. 294–311. Citado 2 vezes nas páginas 8 e 10.

8 CHOW, Y. W.; SUSILO, W. Text-based captchas over the years. *IOP Conference Series: Materials Science and Engineering*, v. 273, n. 1, p. 012001, 2017. Disponível em: ⟨http://stacks.iop.org/1757-899X/273/i=1/a=012001⟩. Citado 2 vezes nas páginas 8 e 15.

9 GOODFELLOW, I. J. et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013. Disponível em: ⟨http://arxiv.org/abs/1312.6082⟩. Citado 3 vezes nas páginas 8, 11 e 12.

10 GEORGE, D. et al. A generative vision model that trains with high data efficiency and breaks text-based captchas. v. 358, p. eaag2612, 10 2017. Citado na página 8.

11 BURSZTEIN, E. et al. The end is nigh: Generic solving of text-based captchas. In: *WOOT*. [S.l.: s.n.], 2014. Citado na página 8.

12 CHELLAPILLA, K. et al. *Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs)*. [S.l.]: Springer, Berlin, Heidelberg, 2005. v. 3517. 1-26 p. Citado 5 vezes nas páginas 10, 11, 12, 15 e 16.

13 SECURIMAGE. *An open-source free PHP CAPTCHA script f*. 2018. Disponível em: ⟨https://www.phpcaptcha.org/⟩. Acesso em: 23/06/2018. Citado na página 10.

14 AHN, L. von et al. recaptcha: Human-based character recognition via web security measures. v. 321, p. 1465–8, 09 2008. Citado na página 11.

15 SIVAKORN, S.; POLAKIS, I.; KEROMYTIS, A. D. I am robot: (deep) learning to break semantic image captchas. In: . [S.l.]: IEEE, 2016. p. 388–403. Citado na página 12.

16 YAN, J.; AHMAD, A. S. E. Breaking visual captchas with naive pattern recognition algorithms. p. 279–291, 01 2008. Citado na página 12.

17 SIMPLECAPTCHA. *A CAPTCHA Framework for Java*. 2018. Disponível em: ⟨http://simplecaptcha.sourceforge.net/⟩. Acesso em: 23/06/2018. Citado na página 16.

18 HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. Disponível em: ⟨http://arxiv.org/abs/1502.01852⟩. Citado na página 17.

19 KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. Disponível em: ⟨http://arxiv.org/abs/1412.6980⟩. Citado na página 17.

20 PRECHELT, L. Automatic early stopping using cross validation: Quantifying the criteria. v. 11, p. 761–767, 06 1998. Citado na página 18.