Relatório Final

Processos atencionais e aprendizado de máquina para sistemas robóticos

Aluno: Erik de Godoy Perillo (RA 135582)
Orientadora: Profa. Dra. Esther Luna Colombini

Instituto de Computação
Universidade Estadual de Campinas

14 de agosto de 2017

1 Introdução

A capacidade de percepção e construção de um modelo da realidade ao seu redor é fundamental para que sistemas robóticos interajam com o ambiente e executem tarefas diversas e complexas que podem ter as mais variadas utilidades para os humanos. Dentro da percepção, a visão é um dos sentidos mais importantes para diversos seres vivos (em especial aos humanos).

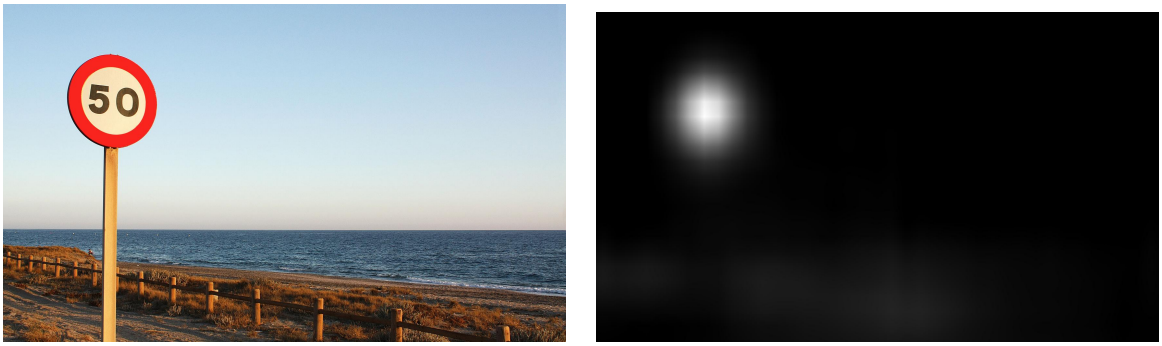
Um problema do sensoriamento é que nossos cérebros, a cada segundo, recebem uma quantidade imensa de informação. Processar todos os estímulos é uma tarefa inviável. Um componente fundamental para lidar com esse problema é a atenção: a habilidade de dar foco apenas ao relevante, evitando assim o processamento desnecessário de enormes quantias de dados. É razoável inspirar-se na atenção para a construção de mecanismos semelhantes que sirvam a sistemas de inteligência artificial em máquinas.

Um conceito importante na atenção visual é a distinção *Bottom-up vs. Top-down*: Por componente *bottom-up* de atenção entende-se saliências instintivas percebidas por mudanças e/ou contrastes muito grandes em uma cena. O componente *top-down* é aquele que dá saliência variável às *features* de acordo com a meta do agente do momento. Neste trabalho, foca-se na atenção *bottom-up*, também chamada de saliência visual.

A atenção tem sido foco de estudo há anos, resultando em diversas teorias em psicologia sobre a atenção humana que inspiraram a implementação de modelos computacionais bem sucedidos. Duas das mais famosas teorias da psicologia sobre atenção visual são a *Feature Integration Theory* (FIT) [15] e a *Guided Search* [5]. Ambas provêm contribuições importantes para o entendimento dos processos de saliência visual e diversos modelos computacionais foram criados baseando-se em ideias delas. A FIT indica basicamente que se a busca de um objeto de interesse em uma cena for por apenas uma *feature*, a localização é feita em tempo instantâneo. Entretanto, se o objeto de interesse for composto por múltiplas *features* a serem buscadas (e. g. uma linha horizontal verde), a localização do objeto é feita em tempo linear. Já *Guided Search* diz que buscas por conjunções de *features* são na verdade mais rápidas pois a combinação das features gera um sinal de saliência mais forte no campo visual humano.

1.0.1 Mapas de saliência

Um conceito importante no contexto de saliência visual são os mapas de saliência (figura 1). Para cada imagem, o seu respectivo mapa de saliência é um mapa onde regiões mais chamativas para humanos na imagem original são mais claras, e regiões menos chamativas são mais escuras. Tais mapas são computados a partir de observações de seres humanos feitas nas imagens.



(a) (b)

Figura 1: Exemplo de mapa de saliência. b) é o mapa de saliência onde regiões mais brancas representam áreas mais salientes para humanos na imagem original a).

1.1 Objetivos

Neste trabalho, objetivamos construir um modelo atencional eficiente, focando primeiramente na saliência visual. Assim, o modelo deve ser capaz de, dada uma imagem de entrada qualquer, produzir um mapa de saliência como o da figura 1, que seja o mais fiel possível a um mapa que seria gerado por seres humanos. O objetivo é que o modelo possa ser usado em um *framework* atencional para robôs móveis em conjunto com outros mecanismo de atenção sensorial.

Em resumo, as principais atividades do trabalho envolvem:

- Revisão bibliográfica sobre teorias sobre a atenção, diversos modelos e métricas.
- Implementação de um modelo atencional com métodos tradicionais.
- Implementação de um modelo atencional com *Deep Learning*, com eficiência suficiente para posterior uso em robôs móveis.

As atividades desenvolvidas são mais detalhadas no decorrer do documento.

2 Materiais e métodos

2.1 Avaliação de desempenho

Foi feita uma pesquisa das métricas e *datasets* mais usados para avaliação dos modelos computacionais, pois isso é muito importante para o avanço do trabalho. As métricas expressam o quão similares são os mapas de saliência gerados por modelos computacionais em relação ao dos seres humanos (*ground-truth*).

2.1.1 *Datasets*

Os *datasets* são compostos de imagens juntamente ao conjunto-verdade, algo que indica como os mapas de saliência para cada imagem deveriam ser.

Dois dos tipos principais de mapas-verdade:

1. Pontos de fixação de olhar: Algum dispositivo de *eye-tracking* é usado em pessoas para determinar as regiões na imagem onde as mesmas olham em um intervalo de tempo. Apenas os pixels com fixação são brancos.

2. Mapas contínuos: As regiões salientes têm valores contínuos em um intervalo de intensidade de *pixels*, com valores maiores representando uma saliência maior. São geralmente obtidas através dos pontos de fixação, aplicando-se uma gaussiana adequada em cada ponto.

Alguns *datasets* utilizados geralmente:

- `CAT2000 [1]:` 4000 imagens de 20 categorias com dados de fixação de olhar de 24 pessoas por imagem de duração de 5 segundos.
- `JUDD [8]:` 1003 imagens de diversos tipos com dados de fixação de olhar de 15 pessoas por imagem. Acompanha também mapas contínuos. É amplamente usado.
- `MIT300 [9]:` 300 imagens de diversos tipos com dados de fixação. Entretanto, os mesmos não estão disponíveis e devem ser testados pelos donos do *dataset*.
- `SALICON` [6]: 15000 imagens de diversos tipos com dados de fixação e contínuos.

2.1.2 Métricas

Muitas das métricas aqui mostradas são discutidas em [2], onde a aplicabilidade, significado, vantagens e desvantagens de cada uma são discutidas mais a fundo.

- `AUC-Judd`
  Abreviação de *Area Under ROC Curve*. É feita uma varredura em diversos valores de *threshold* para o mapa de saliência e, para cada valor, calcula-se o *true positive rate* e o *false positive rate* com base na máscara feita para o mapa de saliência e os pixels brancos respectivos aos pontos de fixação de humanos. A área embaixo da *ROC* formada então é calculada. Usado em [9, 8].
- `NSS`
  `AUC` pode avaliar com altas pontuações mapas com falsos positivos mas com valores baixos nestes falsos positivos [2]. O `NSS` penaliza isso e dá altas pontuações a mapas com valores baixos em negativos e altos em positivos:

  $$NSS(P,Q) = \frac{1}{N}\sum_{i=1}^{N} \bar{P}_i Q_i$$

  Onde N é o número de pontos de fixação, Q é o mapa binário de pontos de fixação e $\bar{P}$ é o mapa de saliência normalizado pelo desvio padrão. Usado em [9].
- `Similarity`
  Métrica para mapas contínuos. Definido como:

  $$SIM(P,Q) = \frac{1}{N}\sum_{i=1}^{N} min(P_i, Q_i)$$

  Com P e Q indo de 0 a 1. Um valor de 1 define mapas idênticos e 0 mapas totalmente diferentes. Ambos os mapas são normalizados pela soma de cada um. Usado em [9].

- `Correlation Coefficient`
  Métrica para mapas contínuos. `Similarity` penaliza *false negatives* mais que *false positives* [2]. A métrica `CC` trata os dois simetricamente. Dada por:

$$CC(P,Q) = \frac{cov(P,Q)}{\sigma(P)\sigma(Q)}$$

  Onde P é normalizado no intervalo $[0,1]$. Usado em [9].

Para *ground-truth* por pontos binários de fixação, `AUC` é razoável para casos em que se quer um bom *recall* e que falsos negativos mas com valores relativamente baixos não importam muito. Isso é verdade em casos em que o foco atencional será direcionado apenas à região de maior valor. `NSS` também pode ser válido pois ele penaliza falsos positivos mesmo que baixos, focando mais em *precision*. Para *ground-truth* contínuos derivados de pontos de fixação, `SIM` é bom quando importa-se com *recall*, pois a métrica penaliza falsos negativos mais que falsos positivos. `CC` também é boa por ser neutra e penalizar falsos positivos/negativos igualmente.

2.1.3 MIT300 benchmark

O *MIT300 benchmark* [9] é, além de um dataset, um *benchmark* amplamente utilizado para a avaliação do desempenho de modelos de saliência visual. Diversas métricas são utilizadas para a ordenação dos modelos em um *ranking*. É nele que os modelos aqui desenvolvidos serão avaliados.

2.2 Modelo atencional com técnicas tradicionais

O VOCUS [4] é um modelo atencional computacional para a detecção de saliências visuais. A maioria dos seus componentes é feita com base nas ideias da FIT. Seu componente *bottom-up*, ilustrado na figura 2, lida com as *features*: cor, intensidade, orientação. Seus mapas de saliência são calculados com base nessas *features* e em diversas dimensões da imagem. Muitos mecanismos foram inspirados no VOCUS, lidando com as mesmas *features* e em múltiplas dimensões da imagem.

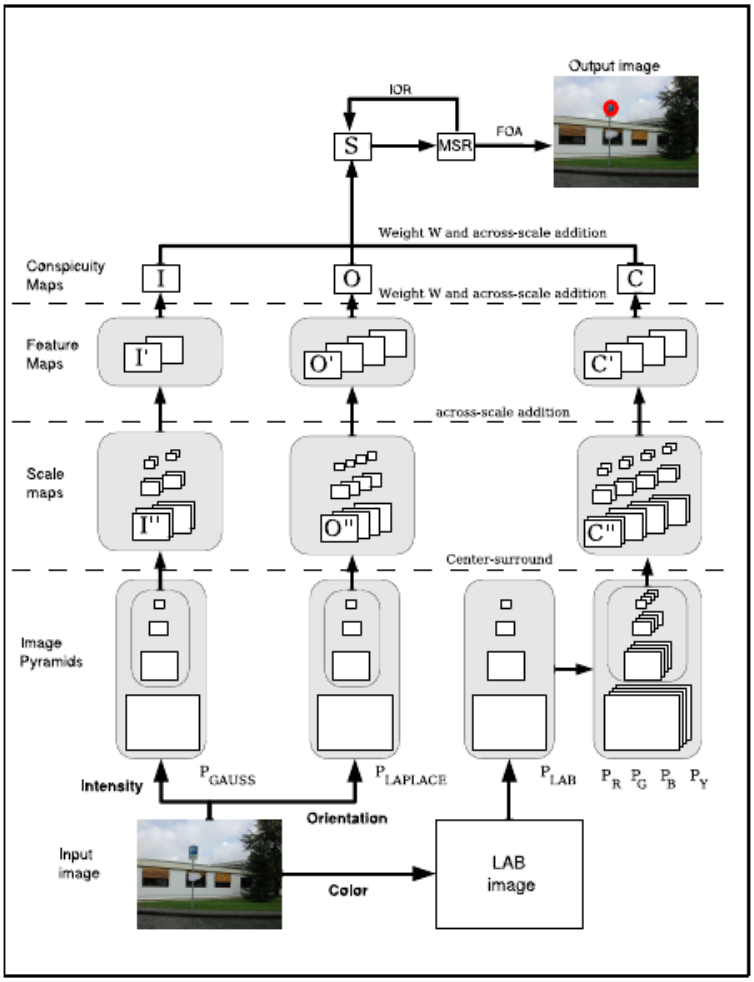


Figura 2: Componente *bottom-up* do VOCUS.

2.2.1 Extração de *features*

O modelo extrai os seguintes mapas de uma certa imagem: luminância, luminância invertida, vermelho, verde, amarelo, azul, orientações vertical, horizontal, 45°e 135°. Os de luminância e cor podem ser extraídos pela conversão da imagem para o espaço de cor LAB e as orientações são extraídas usando-se filtros de Gabor.

2.2.2 Extração de saliência

Para cada mapa, a saliência é calculada usando-se o mecanismo de *center-surround* [4]: uma operação que basicamente extrai contrastes fortes do mapa, dando intensidades de pixel altas para essas regiões. O *center-surround* pode ser aplicado por meio da convolução de um *kernel* apropriado na imagem. Tal *kernel* pode ter tamanho variado, mas a soma de seus valores deve ser zero, o centro deve ter positivo e a vizinhança deve ter valores negativos.

Figura 3: À esquerda, a imagem original. No centro, seu mapa de oponência amarelo-azul. À direita, o resultado de *center-surround* no mapa de cor.

2.2.3 Mapas de saliência para cada *feature*

Para cada *feature* (e. g. vermelho) é calculado o *center-surround*. Isso é feito na imagem original e em diversas outras dimensões dela, calculando-se a pirâmide da imagem. Geralmente usa-se quatro níveis. Isso é importante para capturar saliências nos mais diversos níveis de detalhe da imagem. Uma vez calculados, todos os mapas de uma certa *feature* são redimensionados para as dimensões originais e somados, formando assim um mapa de *feature*.

2.2.4 Normalização

Uma vez calculados os mapas para cada *feature* (vermelho, orientação horizontal etc), é preciso fazer uma normalização nos mesmos. Isso se deve ao fato que, se há grande frequência de picos de saliência no mapa de vermelho, por exemplo, este não é de muito valor, pois o que se quer identificar são regiões salientes com relação à imagem como um todo. Assim, para cada mapa é calculado um peso de normalização.

São diversos os critérios desenvolvidos, como: número de máximos locais, densidade de máximos locais, espalhamento espacial dos máximos. Uma análise das alternativas não mostrou muitas diferenças no desempenho, então opta-se pelo método mais simples, por padrão, que é o número de máximos locais. Isso pode ser obtido por limiar *Otsu*, seguido de um algoritmo de componentes conexos para contar os máximos locais.

2.2.5 Mapas de saliência para cada *feature*

Uma combinação hierárquica dos mapas é feita após as normalizações. No final dessa operação, há três mapas: cor, luminância e orientação. Eles são formados simplesmente somando e normalizando suas instâncias: O de cor, por exemplo, é formado somando-se os mapas de saliência de vermelho, verde, amarelo e azul.

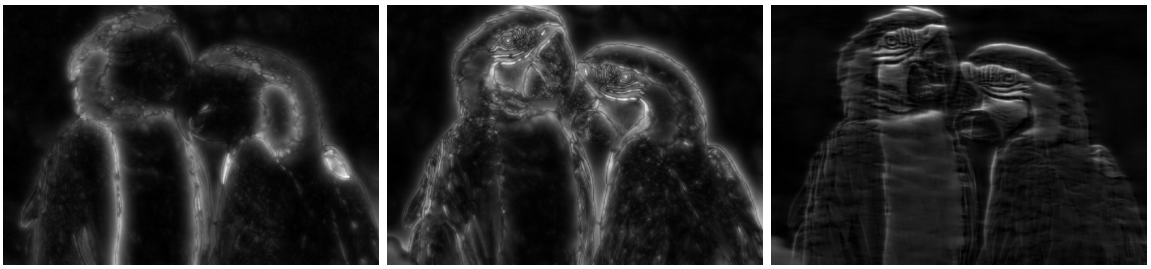


Figura 4: Mapas de saliência da figura original 3. Esquerda: mapa de cor. Centro: mapa de luminância. Direita: mapa de orientação.

2.2.6 Combinação final

É dado um peso para cada mapa de saliência (cor, luminância, orientação) e então eles são somados e normalizados. Nos testes feitos, obteve-se melhores resultados dando peso maior para a cor e menor para a orientação. Nos exemplos aqui, os pesos são 2, 1, 0.1 para cor, luminância e orientação, respectivamente.

2.2.7 Implementação

Todas as etapas do modelo descritas aqui foram implementadas. A linguagem utilizada foi *Python*, usando-se *OpenCV* e *numpy*. O código está disponível de modo público [12].

2.3 Modelo atencional com *Deep Learning*

A tarefa de detecção de saliência visual torna-se especialmente difícil em imagens normais não sintéticas, onde há diversos componentes potencialmente chamativos e relações complexas de contraste e cor. Um significativo aumento no desempenho dos modelos ocorreu quando começou-se a usar técnicas de *Deep Learning* para a tarefa.

Modelos como *Salicon* [6] demonstraram que o uso de redes neurais convolucionais com pesos inicializados de redes utilizadas para classificação como a *VGG-16* [13] poderiam levar a mapas de saliência visual muito similares a aqueles gerados por humanos. *ML-Net* [3] usa a saída de diferentes camadas da *VGG-16*, combinado-as em várias dimensões e diferentes níveis de abstração. *DeepFix* [7] extende um modelo pré-treinado com novos layers que consideram *features* globais e *center bias*. *Salnet* [10] explora dois modelos que são simples mas dão bons resultados.

O grande problema dos modelos atuais que usam *Deep Learning* é que estes são custosos, em geral porque muitos usam redes de tarefas que não originalmente são detecção de saliência visual, como classificação. Tais redes são feitas para classificar milhares de objetos, o que requer muitos parâmetros. Assim, busca-se um modelo que seja eficiente, tendo bons resultados e sendo menos computacionalmente custoso. Para esta tarefa, exploramos diversos métodos de pré-processamento de dados e uma arquitetura de rede neural totalmente convolucional que fossem relevantes no contexto de saliência visual. O modelo final foi nomeado *DeepPeek*.

2.3.1 Arquitetura proposta

A figura 2.3.1 mostra a arquitetura da rede neural totalmente convolucional proposta no trabalho. A rede extrai *features* de dimensões cada vez menores da imagem de entrada.

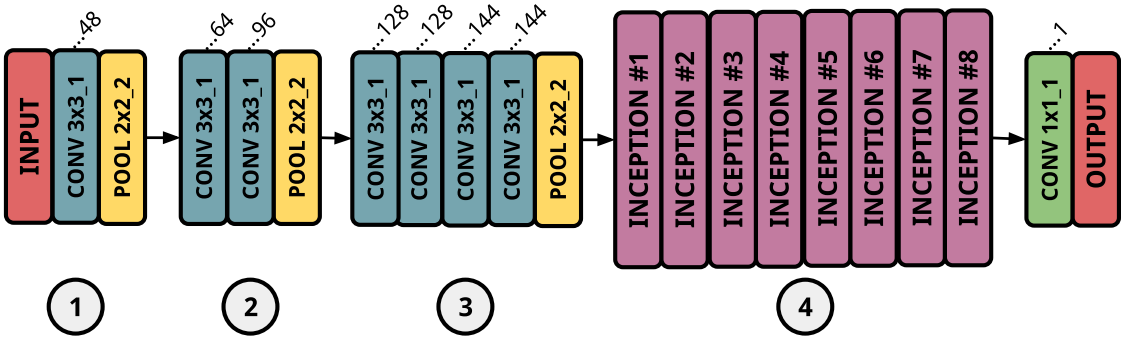


Figura 5: Visão geral da arquitetura proposta. Tamanhos de filtros estão em formato largura×altura_passo.

A rede é composta de quatro blocos principais:

1. O primeiro nível extrai *features* de baixo nível de abstração da imagem de entrada, de dimensões $W \times H \times 3$ (largura, altura, canais), usando uma única camada com 48 filtros de convolução com ativação ReLU seguida por *max-pooling* que reduz a imagem por um fator de dois. Reduzir o número de filtros nesta camada mostrou-se prejudicial para o desempenho da rede, pois é importante captar informação de alta frequência no contexto de saliência visual.

2. O segundo nível extrai *features* de baixo/médio nível de abstração da entrada de dimensões $W/2 \times H/2 \times 48$ usando duas camadas com 64 e 96 filtros de convolução, respectivamente, seguidos por ativação ReLU e *max-pooling*.

3. O terceiro nível extrai *features* de médio/alto nível de abstração da entrada de dimensões $W/4 \times H/4 \times 96$ usando quatro filtros de convolução. As primeiras duas camadas têm 128 filtros cada e as duas últimas têm 144 filtros cada. Toda camada de convulução é seguida de ReLU. *Max-pooling* acontece no final. Um grande número de filtros neste nível mostrou-se importante para o desempenho da rede.

4. O quarto e último nível é composto de oito blocos *inception* que extraem *features* de alto nível de abstração da entrada com dimensões $W/8 \times H/8 \times 144$. Um número elevado de blocos mostrou-se importante. Uma convulução de 1×1 no final faz uma combinação linear dos mapas de saída dos blocos *inception*, seguida por ReLU, produzindo o mapa final de saliência de dimensões $W/8 \times H/8 \times 1$. O mapa é redimensionado para as dimensões originais da imagem com interpolação bicúbica.

A figura 6 ilustra a arquitetura *inception* [14] usada em cada bloco onde são aplicados filtros de dimensões 5×5, 3×3 (ambos precedidos por convoluções 1×1 para redução de dimensionalidade) , 1×1, e *max-pooling* de tamanho 3×3. Cada um desses filtros é aplicado em paralelo da mesma entrada e as saídas são concatenadas no final. *Inception* permite que a rede use informação de diferentes resoluções espaciais assim como de camadas anteriores (mais baixo nível), aspectos importantes no contexto de saliência visual.

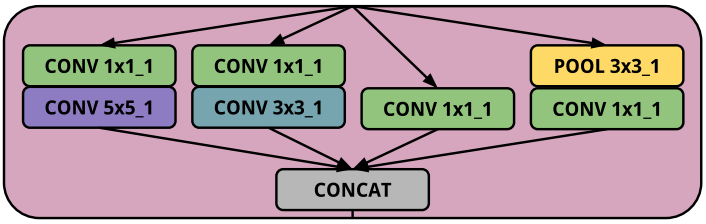


Figura 6: Bloco *inception*

.

A rede tem um total de 3717841 parâmetros, um número muito baixo comparado a outros modelos. A tabela 2.3.1 detalha os números de filtros para cada bloco *inception* utilizado.

Tabela 1: Número de filtros usado em cada bloco *inception*.

Bloco	pool	conv 1×1	3×3 reduce	conv 3×3	5×5 reduce	conv 5×5
1	96	128	96	192	58	96
2	64	128	80	160	24	48
3	64	128	80	160	24	48
4	64	128	96	192	28	56
5	64	128	96	192	28	56
6	64	128	112	224	32	64
7	64	128	112	224	32	64
8	112	160	128	256	40	80

2.3.2 Implementação

A rede foi implementada usando-se *Theano* `0.9.0.dev` com *Lasagne* `0.2.dev1` em uma máquina com *Ubuntu 16.04 LTS* e kernel *Linux* `4.8.0-54-generic`. O treinamento foi feito em uma GPU *NVIDIA GTX 1080* e o código está disponível em `https://goo.gl/WzpyYJ`.

2.3.3 Treinamento

Durante o treinamento, as imagens foram redimensionadas para $320 \times 240 \times 3$. Cada imagem é normalizada por canal pela subtração da média e divisão pelo desvio padrão. Além disso, dois aspectos são importantes nesta etapa: O uso de normalização por imagem, ao invés de pelo *dataset*, e a conversão do espaço de cor das imagens para LAB (ao invés do comum RGB). Esses procedimentos não foram encontrados na literatura, mas são consideradas importantes no contexto de saliência visual. A normalização por imagem usa mais informação do contexto global da imagem, e *VOCUS* cita que o espaço de cor LAB é mais fiel ao sistema visual humano uma vez que ele contém mapas vermelho-verde, amarelo-azul e mapas de luminância. Conjectura-se que tais mapas facilitam na extração de *features* importantes. Experimentos iniciais mostraram que as duas técnicas resultam em melhor desempenho da rede.

A função objetivo a se minimizar foi o coeficiente de correlação entre o mapa de saliência *ground-truth* G e o mapa previsto P:

$$CC(P, G) = \frac{cov(P, G)}{(\sigma(P)\sigma(G))}$$

Tal função de custo é considerada apropriada pois ela penaliza simetricamente falsos positivos e negativos.

Dois *datasets foram utilizados*: *SALICON* e *Judd*. A rede foi treinada usando-se *Stochastic Gradient Descent* com Momento de *Nesterov* de 0.9. Salicon foi primeiramente usado com *data augmentation* espelhando-se imagens horizontalmente e verticalmente. O treinamento ocorreu por 5 épocas com *learning rate* de 0.009 e então por 3 épocas com *learning rate* de 0.001. Mudou-se então para o *dataset* Judd. O treinamento ocorreu por mais uma época, com *learning rate* de 3×10^{-5} e regularização L2 de 10^{-4}. Os tamanhos de batch foram 10 para SALICON e 2 para Judd. O processo de treinamento todo levou cerca de duas horas e meia.

3 Resultados

As figuras 7 e 8 ilustram, respectivamente, mapas gerados pelos modelos *att* (tradicional) e *DeepPeek* (com *Deep Learning*). A tabela 3 mostra os resultados obtidos pela submissão dos modelos ao *MIT300 benchmark* em comparação com alguns dos melhores modelos do ranking.

Figura 7: Mapa de saliência final para uma figura gerado pelo modelo *att*. À esquerda, a imagem original. À direita, o mapa de saliência final.

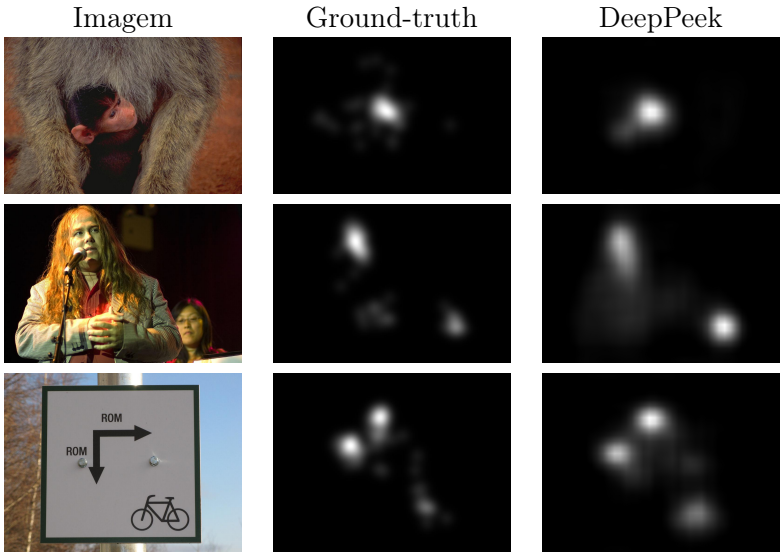


Figura 8: Exemplos de mapas de saliência gerados pelo modelo *DeepPeek*.

Tabela 2: Comparação de resultados com modelos propostos e melhores modelos no *MIT300 benchmark*.

Modelo	Num. parâmetros	AUC-Judd ↑	CC ↑	NSS ↑	Sim ↑	EMD ↓
DeepFix	≈16.7 milhões	0.87	0.78	2.26	0.67	2.04
Salicon	≈14.7 milhões	0.87	0.74	2.12	0.60	2.62
DeepPeek	**3.72 milhões**	**0.85**	**0.71**	**1.98**	**0.62**	**2.37**
ML-Net	≈15.4 milhões	0.85	0.69	2.07	0.60	2.53
SalNet	25.8 milhões	0.83	0.57	1.51	0.52	3.31
att	-	**0.62**	**0.12**	**0.34**	**0.33**	**5.28**

Nota-se que o *DeepPeek* obteve resultados muito similares aos dos melhores modelos atuais, com cerca de um quarto do número de parâmetros dos outros modelos.

4 Conclusões

Neste trabalho, foram explorados dois modelos de saliência visual: *att*, um modelo baseado no modelo VOCUS, que usa técnicas de extração de um conjunto pré-selecionado de *features*; e *DeepPeek*, uma rede neural totalmente convolucional com foco em eficiência.

A tabela 3 mostra que o modelo *att* tem desempenho muito abaixo dos melhores modelos atuais. Os resultados indicam que as técnicas atuais de aprendizado de filtros de extração de *features* são de fato necessárias para um bom desempenho na tarefa de detecção de saliência visual.

O modelo *DeepPeek* mostra um bom desempenho, sendo comparável ao dos melhores modelos atualmente, tendo apenas cerca de 1/4 do número de parâmetros dos outros modelos. Assim, conclui-se que um modelo eficiente foi obtido com sucesso. O desenvolvimento de uma arquitetura completamente dedicada à saliência visual, juntamente com técnicas de pré-processamento de dados especialmente para o contexto de saliência, mostraram-se importantes para o resultado obtido.

4.1 Próximos passos

Planeja-se, em futuros trabalhos, estender o modelo *DeepPeek* para que o mesmo seja usado para a detecção de saliência visual em vídeos. Tal domínio ainda é pouco explorado por modelos atuais e requer a consideração de aspectos novos. Saliência em vídeo é importante para o uso do sistema por robôs móveis.

5 Publicações derivadas do trabalho

O trabalho envolvido na obtenção do modelo *DeepPeek* resultou em um artigo, *Efficient Visual Attention with Deep Learning* [11], publicado no WTD2017 [16] do Instituto de Computação da Unicamp. O artigo e a apresentação do mesmo resultaram na premiação de **melhor trabalho de Iniciação Científica** do WTD2017.

Referências

[1] Ali Borji e Laurent Itti. "CAT2000: A Large Scale Fixation Dataset for Boosting Saliency Research". Em: *CVPR 2015 workshop on "Future of Datasets"* (2015).

[2] Zoya Bylinskii et al. "What do different evaluation metrics tell us about saliency models?" Em: (2016).

[3] Marcella Cornia et al. “A Deep Multi-Level Network for Saliency Prediction”. Em: *arXiv preprint arXiv:1609.01064* (2016).

[4] Simone Frintrop. “Vocus: a visual attention system for object detection and goal-directed search”. Tese de doutorado. 2006.

[5] Susan L. Franzel Jeremy M. Wolfe Kyle R. Cave. “Guided Search: an alternative to the feature integration model for visual search”. Em: (1989).

[6] Ming Jiang et al. “Salicon: saliency in context”. Em: *CVPR* (2015).

[7] Srinivas S S Kruthiventi, Kumar Ayush e R. Venkatesh Babu. “DeepFix: A Fully Convolutional Neural Network for predicting Human Eye Fixations”. Em: *arXiv preprint arXiv:1510.02927* (2015).

[8] *Learning to predict where people look*. 2016. URL: `http://people.csail.mit.edu/tjudd/WherePeopleLook/index.html` (acesso em 04/10/2016).

[9] *MIT saliency benchmark*. 2016. URL: `http://saliency.mit.edu/index.html` (acesso em 27/09/2016).

[10] Junting Pan et al. “Shallow and Deep Convolutional Networks for Saliency Prediction”. Em: *arXiv preprint arXiv:1603.00845* (2016).

[11] Erik Perillo e Esther Colombini. *Efficient Visual Attention with Deep Learning*. 2017. URL: `https://drive.google.com/file/d/0B1VvuVdhIVJnTUhwYk9KM2d5a2c/view` (acesso em 13/08/2017).

[12] *Repositório att*. 2016. URL: `https://github.com/erikperillo/att` (acesso em 09/12/2016).

[13] Karen Simonyan e Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Em: *CoRR* abs/1409.1556 (2014). URL: `http://arxiv.org/abs/1409.1556`.

[14] Christian Szegedy et al. “Going Deeper with Convolutions”. Em: *CoRR* abs/1409.4842 (2014). URL: `http://arxiv.org/abs/1409.4842`.

[15] A M Treisman e G Gelade. “A feature-integration theory of attention”. Em: *Cognit Psychol* 12.1 (jan. de 1980), pp. 97–136.

[16] *XII Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica*. 2017. URL: `http://www.ic.unicamp.br/wtd/2017/` (acesso em 13/08/2017).