

UNIVERSIDADE FEDERAL FLUMINENSE

HEVERTON DE MELO SARAH

**SEGMENTAÇÃO DE BRAÇOS E MÃOS PARA
PERSPECTIVA EGOCÊNTRICA BASEADA EM
PSPNET E DEEPLAB**

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

HEVERTON DE MELO SARAH

SEGMENTAÇÃO DE BRAÇOS E MÃOS PARA PERSPECTIVA EGOCÊNTRICA BASEADA EM PSPNET E DEEPLAB

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: COMPUTAÇÃO VISUAL

Orientador:

Esteban Walter Gonzalez Clua

Co-orientador:

Cristina Nader Vasconcelos

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S243s Sarah, Heverton de Melo
Segmentação de braços e mãos para perspectiva
egocêntrica baseada em PSPNet e DeepLab / Heverton de Melo
Sarah ; Esteban Walter Gonzalez Clua, orientador ; Cristina
Nader Vasconcelos, coorientadora. Niterói, 2019.
67 f. : il.

Dissertação (mestrado)-Universidade Federal Fluminense,
Niterói, 2019.

DOI: <http://dx.doi.org/10.22409/PGC.2019.m.00819281220>

1. Segmentação de imagem. 2. Realidade virtual. 3.
Computação visual. 4. Videogame. 5. Produção intelectual.
I. Gonzalez Clua, Esteban Walter, orientador. II. Vasconcelos,
Cristina Nader, coorientadora. III. Universidade Federal
Fluminense. Instituto de Computação. IV. Título.

CDD -

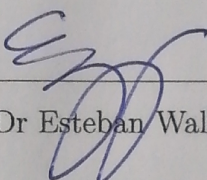
HEVERTON DE MELO SARAH

Segmentação de braços e mãos para perspectiva egocêntrica baseada em PSPNet e Deeplab

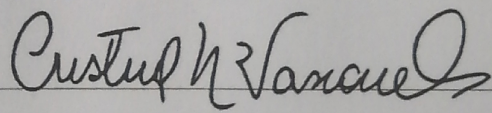
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Computação Visual.

Aprovada em Julho de 2019.

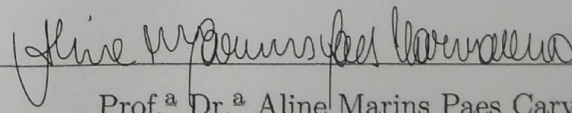
BANCA EXAMINADORA



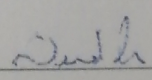
Prof. Dr. Esteban Walter Gonzalez Clua - Orientador
UFF



Prof.ª Dr.ª Cristina Nader Vasconcelos - Co-orientador
UFF



Prof.ª Dr.ª Aline Marins Paes Carvalho
UFF



Prof.ª Dr.ª Veronica Teichrieb
UFPE

Niterói
2019

Dedico à Moana, a que mais entende a importância desse momento.

Agradecimentos

Agradeço ao professor Esteban por ter me orientado durante a pesquisa e toda a jornada do mestrado, que começou em um evento de tecnologia no Acre. Foi responsável por me dar a certeza de que atuar na indústria de games é um sonho possível de realizar.

Agradeço à professora Cristina por me auxiliar até mesmo com filho no colo e longe do Brasil. Foi responsável por me apresentar essa área de estudos que não para de surpreender no que é possível fazer.

Tenho muito a agradecer à Elizete por me acompanhar em todos os momentos dessa batalha, a pessoa que mais sabe sobre o que passei desde o início da graduação, e responsável por eu não ter desistido em nenhum momento.

Agradeço aos meus pais por suportarem a minha distância e ausência por motivo de estudo e muito trabalho.

Agradeço também ao Leonardo Torok, que infelizmente não está mais entre nós, mas que me ajudou muito durante a época complicada de disciplinas no mestrado.

Resumo

Vídeos em primeira pessoa e jogos são os paradigmas centrais de posicionamento de câmera quando Head Mounted Displays (HMDs) são utilizados. Nessas situações, as mãos e braços do usuário possuem um papel fundamental no sentimento de auto-presença e interface. Enquanto as representações visuais de braços e mãos são triviais em dispositivos de Realidade Aumentada ou quando câmeras de profundidade são conectadas em HMDs, suas renderizações não são simples de serem resolvidas com simples HMD, como os baseados em smartphones. Este trabalho propõe o uso de segmentação semântica com Redes Neurais Totalmente Convolucionais para separar as mãos e braços do usuário de uma imagem, capturada por câmeras comuns posicionadas em um esquema de perspectiva em primeira pessoa. Primeiramente, criamos uma base de imagens para treinamento composta por 4041 imagens e outra para validação com 322, ambas com máscaras indicando pixels que pertencem à braços ou não-braços, focadas na perspectiva de primeira pessoa. Então, baseado em duas arquiteturas importantes para segmentação semântica - PSPNet e DeepLab - propomos uma calibração para o cenário particular composto de mãos e braços, capturados em perspectiva de HMD. Nossos resultados mostram que PSPNet possui melhor detalhe de segmentação enquanto DeepLab alcança melhor tempo de inferência. O treinamento com nossa base em visão egocêntrica gera melhores resultados de segmentação de braços do que o treinamento com imagens em diferentes e mais gerais perspectivas.

Palavras-chave: Segmentação de imagem de pele egocêntrica, Redes Neurais Totalmente Convolucionais, Headset de Realidade Virtual, Auto-Presença.

Abstract

First person videos and games are the central paradigms of camera positioning when using Head Mounted Displays (HMDs). In these situations, the user's hands and arms play a fundamental role in self-presence feeling and interface. While their visual image is trivial in Augmented Reality devices or when using depth cameras attached to the HMDs, their rendering is not simple to be solved with simple HMD, such as those based on smartphone devices. This work proposes the usage of semantic image segmentation with Fully Convolutional Networks for detaching user's hands and arms from a raw image, captured by regular cameras, positioned as a First Person visual schema. We first create a training dataset composed by 4041 images and a validation dataset composed of 322 images, both of them with labels for an arm and no-arm pixels, focused on egocentric view. Then, based on two important architectures related to semantic segmentation - PSPNet and DeepLab - we propose a specific calibration for the particular scenario composed of hands and arms, captured from an HMD perspective. Our results show that PSPNet has better detail segmentation while DeepLab achieves best inference time performance. Training with our egocentric dataset generates better arm segmentation than using images in different and more general perspectives.

Palavras-chave: Egocentric skin image segmentation, Fully Convolutional Networks, Head Mounted Display, Self-Presence.

Lista de Figuras

2.1	Camadas de Rede de Convolução (Fonte: [67])	5
2.2	Exemplo de convolução (Fonte: [31])	7
2.3	<i>Max Pooling</i> (Fonte: [31])	7
2.4	Exemplo de transferência de aprendizado	8
2.5	Arquitetura AlexNet (Fonte: [33])	10
2.6	Módulo ResNet (Fonte: [26])	11
2.7	Módulo Inception (Fonte: [64])	11
2.8	<i>Depthwise Convolutions</i> (Fonte: [71])	12
2.9	<i>Pointwise Convolution</i> (Fonte: [71])	12
2.10	DeepLabV3 aplicada em ResNet (Fonte: [10])	13
2.11	Arquitetura DeepLabv3+ (Fonte: [7])	14
2.12	Arquitetura Xception Modificada (Fonte: [7])	15
2.13	<i>Inverted Residuals</i> (Fonte: [54])	16
2.14	Arquitetura PSPNet (Fonte: [76])	17
2.15	Arquitetura FCN (Fonte: [37])	20
2.16	Matriz de Convolução (Fonte: [16])	20
2.17	Agrupamento e Desagrupamento (Fonte: [75])	21
2.18	Encoder-Decoder da SegNet (Fonte: [2])	22
2.19	Dilatação de filtros (Fonte: [22])	24
2.20	Expansão de Campos Receptivos relacionados com os filtros (Fonte: [73]) .	24
2.21	Gestos da base FPVRGame (Fonte: [46])	26
3.1	Exemplo de Maurya et al. [42]. (Fonte: [42])	28

3.2	Exemplo de Li et al. [35]. (Fonte: [35])	29
3.3	Visão geral de Siddharth et al. [55]: (a) Imagem original (b) Detecção de mãos (c) Classificação de pele a nível de píxel (d) Extração de mão localizada (e) Classificação de intenção do motorista. (Fonte: [55])	29
3.4	Visão geral da arquitetura de Vodopivec et al.. (Fonte: [70])	30
4.1	Exemplos de cada banco de imagens. (A) ViolaSet, (B) SFA, (C) Egohand, (D) EgohandVal.	35
5.1	Resultados Qualitativos. (A) is the original image, (B) ground truth, (C) deeplab_egohand_14b_mobilenet, (D) deeplab_huskin_14b_mobilenet, (E) deeplab_egohand_14b_xception, (F) deeplab_huskin_14b_xception, (G) psp_egohand_no_aug, (H) psp_huskin_no_aug.	43
5.2	Testes na base GTEA. (A) imagem original de 4 sujeitos em atividades diferentes, (B) máscara de braços, (C) PSP, (D) DeepLab.	45

Lista de Tabelas

2.1	Resumo comparativo das principais redes	18
3.1	Resumo comparativo de trabalhos relacionados	31
5.1	Accurácia da PSPNet no conjunto de validação	40
5.2	F-Score da PSPNet no conjunto de validação	41
5.3	MCC da PSPNet no conjunto de validação	41
5.4	Acurácia da Deeplab no conjunto de validação	41
5.5	F-Score da Deeplab no conjunto de validação	42
5.6	MCC da Deeplab no conjunto de validação	42
5.7	Teste de tempo de inferência	44
5.8	Teste na base pública GTEA	44

Lista de Abreviaturas e Siglas

AR	: Augmented Reality;
ASPP	: Atrous Spatial Pyramid Pooling;
CNN	: Convolutional Neural Network;
CRF	: Conditional Random Fields;
FCN	: Fully Convolutional Network;
FPV	: First Person Vision;
GPU	: Graphics Processing Unit;
GTEA	: Georgia Tech Egocentric Activity;
HMD	: Head Mounted Display;
MCC	: Matthews Correlation Coefficient;
MOMF	: Multi Orientation Matched Filtering;
PSPNet	: Pyramid Scene Parsing Network;
ReLU	: Rectified Linear Unit;
VR	: Virtual Reality;

Sumário

1	Introdução	1
2	Referencial Teórico	4
2.1	Redes Neurais de Convolução	5
2.1.1	Conceitos Básicos	5
2.1.2	Redes Importantes	9
2.2	Segmentação de Imagens com Redes de Convolução	17
2.2.1	Encoder-Decoder	22
2.2.2	Baseada no contexto	23
2.2.3	Bases de Imagens	25
2.2.3.1	Pattern Analysis Statistical Modeling and Computational Learning Visual Object Classes (PASCAL VOC)	25
2.2.3.2	FPVRGame	26
2.2.3.3	Georgia Tech Egocentric Activity Datasets (GTEA)	26
2.2.3.4	SFA	26
2.2.3.5	ViolaSet	27
3	Trabalhos Relacionados	28
4	Segmentação de Braços Humanos em Visão Egocêntrica com PSPNet E DeepLab	32
4.1	PSPNet	32
4.2	Deeplab	33
4.3	Configurações dos Treinamentos	33

4.3.1	Bancos de Imagens	33
4.3.2	Configurações de treinamento de DeepLab	35
4.3.3	Configurações de treinamento PSPNet	36
5	Resultados e Discussão	38
5.1	Configuração Experimental	39
5.2	Resultados	40
6	Conclusão	46
	Referências	49

Capítulo 1

Introdução

De acordo com [59] o grau de imersão em um ambiente virtual é relacionado com a descrição da tecnologia, o realismo que um monitor pode mostrar, a quantidade de tipos de sensores, a qualidade ao redor do ambiente virtual e quão panorâmica a realidade virtual é. A imersão requer um corpo virtual, que representa a localização do usuário no ambiente virtual, um aspecto essencial para o conceito de auto-presença. Uma forma de atingir este objetivo consiste em desenhar representações visuais do corpo humano, onde as mãos e braços possuem um papel fundamental, devido a sua constante aparição em ambientes virtuais com perspectiva de primeira pessoa (visão egocêntrica).

A Realidade Virtual (VR) está atualmente presente em diversas aplicações nas quais o usuário experiencia um senso de presença em um ambiente virtual. Atualmente, está mais fácil ter acesso a um Head Mounted Display (HMD) ou *headset* de realidade virtual e experimentar jogos e vídeos em VR, devido ao aumento da atenção que diferentes indústrias estão proporcionando a esta tecnologia. Além disso, câmeras de alta qualidade e portabilidade estão disponíveis à preços acessíveis, o que facilita seu uso em gravação de conteúdo em visão egocêntrica ou primeira pessoa. Esse conteúdo pode ser utilizado para treinar técnicas de detecção automática de braços dos usuários.

A detecção automática e segmentação de pele humana é um tópico de grande interesse devido à larga quantidade de aplicações, como em detecção de rosto humano [69], gestos de mãos [49], detecção de pedestres [15], detecção de pose humana para ambientes de VR [45], dentre muitos outros exemplos. Estas aplicações são o resultado da aplicação de técnicas distintas, em que o processo de separação de píxeis que representam pele em imagens, chamada segmentação de imagem de pele, pode ser uma delas. Neste trabalho, pretendemos segmentar braços e mãos, em tempo real, de imagens em perspectiva egocêntrica.

Redes Neurais de Convolução (do inglês *Convolutional Neural Network* - CNN) são Redes Neurais Profundas que têm mostrado resultados impressionantes em processamento de sinais visuais [67]. A técnica foi inspirada pelo trabalho de [28], que explica que existem neurônios no córtex visual que respondem à regiões pequenas do campo visual, chamadas campos receptivos (do inglês *Receptive Fields*). Cada camada dessas redes são volumes com três dimensões: largura, altura e profundidade. Por exemplo, uma imagem como volume de entrada possui largura e altura, sendo que seus canais de cores são a profundidade do volume. As outras camadas da rede, excluindo a camada de saída, agem como detectores de padrões, onde os padrões simples são detectados nas primeiras camadas e os mais complexos nas últimas. Esses detectores de **características** (do inglês *features*) são somente conectados com pequenas regiões da camada anterior, reduzindo assim a quantidade de parâmetros usados na rede (comparado à quantidade que teria se fosse conectada com toda a região da camada anterior, como acontece em redes neurais convencionais).

Para segmentar uma imagem com uma CNN, pode-se utilizar camadas de convolução desde a entrada da rede até a saída, tornando a rede em uma Rede Totalmente Convolutiva (do inglês *Fully Convolutional Network* - FCN) [38]. Nesse tipo de arquitetura, camadas de *upsampling* aumentam o tamanho espacial de saídas de camadas anteriores, tornando possível realizar predição a nível de píxel quando a rede possui camadas de *subsampling* (que diminuem o tamanho espacial de saídas de camadas anteriores). Além disso, *skip connections* [38] podem ser usadas para unir informações entre camadas que identificam padrões de aparência simples com as que identificam padrões semânticos complexos.

Desde o advento de redes FCN, muitos pesquisadores começaram a mudar outras arquiteturas de redes neurais de convolução para também serem totalmente convolucionais, modificando as últimas camadas totalmente conectadas dessas arquiteturas para convolucionais. Entretanto, o processo de aplicar convoluções diminui o tamanho espacial dos mapas de características (do inglês *feature maps*), o que é bom para obter melhores resultados semânticos nas últimas camadas, mas reduzem a qualidade dos resultados menos semânticos que são obtidos nas primeiras camadas. Este problema fez diferentes autores pensarem em soluções para aumentar o tamanho espacial dos mapas de características ou passar informações das camadas iniciais para as finais. As arquiteturas PSPNet [76] e DeepLab [10, 27, 12] são exemplos de FCNs que resolvem esse problema e que até foram (no caso da PSPNet) ou ainda são (DeepLabv3+ [7]) estado da arte em segmentação semântica na base de imagens pública PASCAL VOC 2012 [20, 18].

Essas redes foram treinadas originalmente para segmentar diversas classes. Para utilizar essas redes neurais para um problema diferente do qual elas foram treinadas, é necessário treiná-las novamente com uma nova base de imagens que represente o novo problema. Para adaptá-las para segmentação de braços e mãos de humanos em perspectiva egocêntrica são necessárias imagens de pessoas nessa perspectiva. Apesar de já existir algumas bases públicas que contenha esse tipo de imagem para segmentação, a quantidade de imagens disponíveis ainda é muito inferior à quantidade disponível para problemas de classificação de imagens. Logo, a construção de mais uma base pode contribuir para que mais pesquisas sejam feitas com quantidade maior de imagens. Outro problema é que essas imagens devem ser capturadas por câmeras posicionadas na região dos olhos do usuário para melhor representar as imagens que serão capturadas na aplicação de realidade mista com visualização de braços do usuário. Outra característica da aplicação é que o usuário irá mover a cabeça, gerando movimento na câmera, ou seja, as imagens de treinamento também devem simular essa situação.

Este trabalho tem como objetivo principal adaptar duas redes de segmentação semântica, PSPNet [76] e DeepLabV3 (com Mobilenet [10], [27] e Xception [12]), para o problema de segmentação de braços e mãos do usuário para aplicações de realidade mista, onde a perspectiva é egocêntrica. Para este objetivo, primeiramente, geramos duas bases de imagens (uma para treinamento e outra para validação) compostas por imagens em perspectiva egocêntrica de braços humanos que representem as situações específicas da aplicação de realidade mista citadas anteriormente. Utilizamos outra base de imagens em variadas perspectivas para confirmar se o treinamento com uma base de imagens em perspectiva egocêntrica melhora a segmentação de imagens em perspectiva egocêntrica. Por último, propomos configurações de ajustes de parâmetros para segmentação específica de braços e mãos, focada em jogos de Realidade Virtual e experiências interativas com posicionamento de câmera no paradigma de visão em primeira pessoa.

Nas próximas sessões mostraremos o referencial teórico relacionado ao trabalho. Na sequência, apresentamos os trabalhos relacionados. Então falamos de como implementamos as configurações nas arquiteturas destacadas para realização de segmentação de braços e mãos humanas. Finalmente, apresentamos nossos experimentos e resultados, seguidos de uma sumarização e descrição de trabalhos futuros.

Capítulo 2

Referencial Teórico

Para visualizar braços do usuário de HMD em aplicações de realidade virtual utilizando segmentação de imagem, a qualidade e velocidade da segmentação são bastante importante para manter a sensação de imersão e tempo real. Ao aplicar redes neurais de convolução para esta tarefa, deve-se direcionar a arquitetura e hiperparâmetros da rede para este objetivo.

Redes Neurais de Convolução possuem elementos que podem ser manipulados para ajustar seu tamanho (camadas, quantidade de filtros e de canais), bem como seu campo receptivo, que como será discutido neste capítulo, é importante para a detecção de detalhes pequenos na segmentação, podendo até detectar o contexto ao qual o objeto a ser segmentado está relacionado. A sessão 2.1 aborda a definição e efeito causado ao ajustar esses componentes. Entender os elementos principais de redes de convolução será importante para observar como foram feitos os experimentos neste trabalho.

A manipulação desses componentes gera diferentes arquiteturas de redes. Algumas das principais e que influenciam as redes utilizadas neste trabalho são abordadas ao final da sessão 2.1. Para a aplicação dessas redes para o problema de segmentação de imagens, camadas de convolução são utilizadas em toda a extensão da rede. A sessão 2.2 aborda como diferentes autores transformaram redes de convolução utilizadas em problemas de classificação para o problema de segmentação através da utilização de camadas de convolução em toda a rede. Ao final da sessão, são apresentadas bases públicas de imagens que foram utilizadas neste trabalho.

2.1 Redes Neurais de Convolução

Redes Neurais de Convolução são redes que utilizam operações de convolução ao invés da multiplicação de matrizes comuns, como ocorre em redes neurais completamente conectadas. Essas operações ocorrem em estruturas de grades, onde os neurônios são organizados. Suas aplicações principais são em processamento de dados visuais, que também é o tipo de processamento que inspirou sua criação nos trabalhos [28] e [29]. Nesta sessão, serão abordados os principais conceitos desse tipo de rede neural e um breve resumo das arquiteturas de redes neurais de convolução mais importantes para entender as que são utilizadas em nossa proposta.

2.1.1 Conceitos Básicos

Na Figura 2.1 pode-se observar que cada camada de uma Rede de Convolução possui uma determinada altura, largura e profundidade. A profundidade é definida pela quantidade de mapas de características que são gerados pela aplicação de filtros por operação de convolução na camada anterior. A primeira camada, quando se trata de uma imagem, possui três canais, onde cada elemento representa uma intensidade do píxel em determinado canal de cor. Cada filtro possui a mesma quantidade de canais da camada em que aplica a operação de convolução, que será detalhada a seguir. Essas redes de convolução possuem três principais tipos de camadas: camadas de convolução, camadas de agrupamento (do inglês *pooling*) e camadas totalmente conectadas.

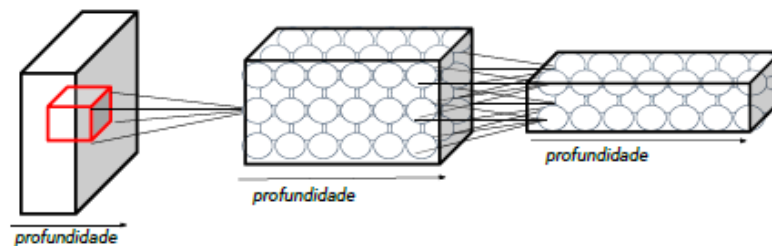


Figura 2.1: Camadas de Rede de Convolução (Fonte: [67])

As camadas de convolução são compostas de filtros que detectam padrões. Quando aplicados às imagens, detectam características das mesmas, tais como: arestas, texturas e outras com significados mais complexos (pele, plano de fundo, etc). As camadas iniciais detectam padrões mais simples e as mais profundas detectam padrões complexos.

Os filtros podem ser aprendidos e possuem mesma profundidade que a entrada da camada. Esses filtros "convolvem" (operação de deslizar pelas dimensões da entrada)

realizando um produto escalar entre os valores do filtro e os valores da entrada na posição atual da convolução, gerando assim um mapa de ativação ou mapa de características de duas dimensões para cada filtro. Esses mapas de ativação definirão a profundidade do volume de saída, que geralmente também passam por uma função de ativação não-linear (sendo a *Rectified Linear Unit* - ReLU a mais utilizada) antes de ser entrada para a próxima camada.

O campo receptivo (do inglês *receptive field*) é a região da entrada de uma camada anterior a uma camada a qual afeta o resultado do mapa da característica resultante desta camada, cujo tamanho (largura e altura) geralmente depende do tamanho do filtro. Como cada camada pode ser afetada por uma parte da anterior, pode-se descer recursivamente até a imagem de entrada da rede e detectar quais pixels afetam cada região de mapas de características de toda a rede até a saída. O tamanho deste campo receptivo pode ser aumentado com a aplicação de múltiplas camadas na rede, agrupamento, uso de passo e outras formas que serão discutidas na sessão 2.2.

Passo (do inglês *Stride*) e *zero-padding* podem ser usados para manipular o tamanho do volume de saída. O passo é a quantidade de píxeis deslizados horizontalmente e verticalmente no volume de entrada, durante a convolução, sendo que quanto maior o passo, menor o tamanho espacial do volume de saída. O *zero-padding* é a quantidade de valores zeros que podem ser colocados ao redor da borda do volume de entrada e é geralmente usado para manter o tamanho do volume de saída semelhante ao de entrada.

Um exemplo da operação do processamento da camada de convolução pode ser visto na Figura 2.2. As fatias de profundidade do volume de entrada (3 canais de cor da imagem de entrada) estão em azul, os filtros da camada de convolução W_0 e W_1 estão em rosa (também em profundidade 3), e volume de saída em cor verde (com profundidade 2, igual a quantidade de filtros da camada de convolução). Pode-se observar também o *zero-padding* igual a 1 (os "zeros" ao redor da borda) ao redor de cada fatia de profundidade do volume de entrada, ou seja, somente uma borda de zeros (seriam duas bordas de "zeros" ao redor, caso o *padding* fosse igual a 2, e assim por diante). O passo é igual a 2, pois a convolução é realizada a cada dois píxeis. O produto escalar é realizado entre cada fatia dos filtros com o mesmo nível de fatia no volume de entrada, somando o resultado desse produto com o de cada fatia, juntamente com o *bias* de cada filtro, gerando assim um elemento em uma fatia do volume de saída, que é relacionada ao filtro da operação. O volume de saída possui duas fatias, a mesma quantidade de filtros que foram aplicados na convolução.

A camada de agrupamento é utilizada para diminuir o tamanho espacial de uma

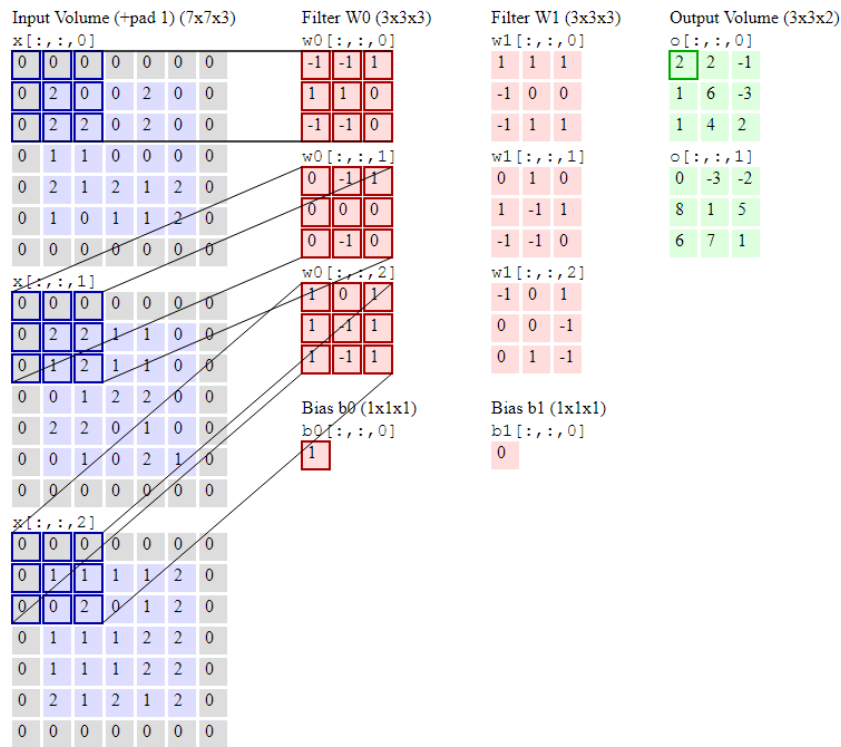
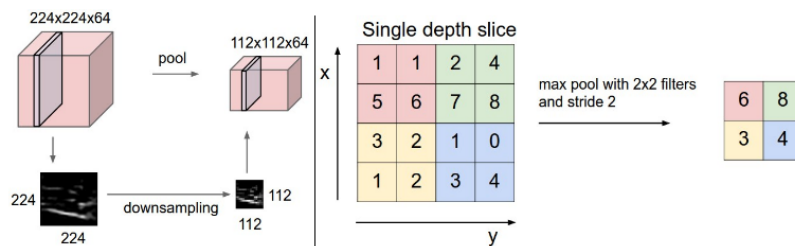


Figura 2.2: Exemplo de convolução (Fonte: [31])

camada de convolução, sem alterar a profundidade do seu volume, para assim diminuir a quantidade de parâmetros e processamento. Para realizar essa operação, um filtro de determinado tamanho desliza sobre toda a extensão das fatias de profundidade da camada de convolução, realizando uma determinada operação entre os elementos vizinhos que estão espacialmente dentro da área do filtro. Essa operação é geralmente a maximização, ou seja, encontra o elemento de maior valor. Na Figura 2.3, pode-se observar uma camada de agrupamento com filtros de tamanho 2x2 e passo 2, onde se realiza a maximização dos elementos da fatia da camada de convolução que estão dentro da área 2x2, com fatia inicial de tamanho 4x4, e resultando 2x2, diminuindo o tamanho inicial. Quando a camada de agrupamento utiliza a operação de maximização, é chamada de *Max Pooling*.

Figura 2.3: *Max Pooling* (Fonte: [31])

Como citado acima, as camadas iniciais de redes de convolução são consideradas mais

genéricas, pois aprendem características mais genéricas, comuns em diferentes tipos de problemas quando a entrada da rede se trata de imagens. Já as últimas camadas aprendem características mais relacionadas com a base em que a rede foi treinada e de acordo com o objetivo da rede. A técnica de transferência de aprendizado (do inglês *transfer learning*) consiste em transferir características que uma rede aprendeu em determinada base de dados para outra rede, que utiliza outra base de dados para treinamento e possui objetivo semelhante ao da anterior ou até diferente. Isso funciona quando se utilizam as características mais genéricas da primeira rede e funciona melhor quando as bases são semelhantes. Entretanto, mesmo quando não são, é possível ver melhora nos resultados [72]. A Figura 2.4 exibe um exemplo de transferência de pesos iniciais de uma rede com arquitetura diferente da rede que recebe os pesos, onde suas camadas iniciais são semelhantes.

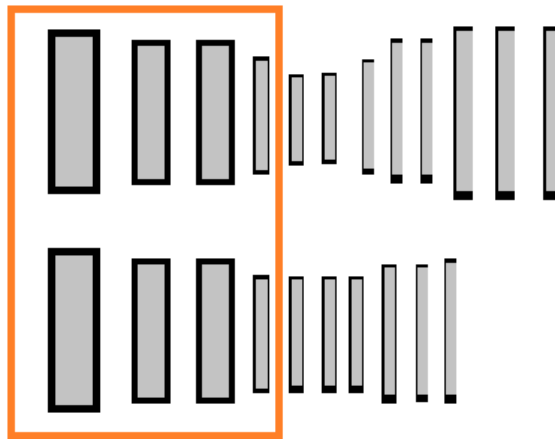


Figura 2.4: Exemplo de transferência de aprendizado

A transferência de aprendizado pode ser utilizada principalmente quando se tem uma base de dados pequena para uma rede com muitas camadas e pesos, que necessitaria de uma base maior para treinar de início ao fim. A forma mais comum de ser implementada é através da cópia das n primeiras camadas de uma rede base para as n camadas da rede para onde se deseja transferir, sendo os pesos dessas camadas da rede base usados para inicializar os pesos das camadas da outra rede e as demais camadas são inicializadas de forma aleatória e treinadas para alguma outra tarefa. A partir dessa arquitetura, pode-se treinar a rede de duas formas: através de *fine-tuning*, que consiste em deixar os erros da nova tarefa se propagarem para as camadas iniciais que foram copiadas da rede base, fazendo com que elas sejam treinadas também; a outra forma é deixar os pesos das primeiras camadas fixos e treinar somente os demais pesos da rede.

Segundo [72], para escolher entre essas duas formas de treinar, é necessário analisar

o tamanho da base que se deseja usar e a quantidade de parâmetros nessas camadas copiadas. Se a base de dados em que se deseja treinar para esta nova tarefa é pequena e o número de parâmetros é grande, ao se realizar *fine-tuning* pode-se introduzir *overfitting*, que consiste no fato de a rede aprender tanto a ponto de que a base de dados não consegue generalizar para outros dados que não sejam idênticos aos que constam na base. Nesse caso, deixar os pesos copiados fixos e treinar somente os demais é a melhor opção. O *fine-tuning* pode ser usado quando a quantidade de parâmetros é pequena ou até mesmo quando a base é grande [72].

Tem sido muito comum a utilização de pesos treinados em redes com bases grandes como a ImageNet [14] para inicializar pesos de outra rede com até mesmo arquitetura diferente da rede base (mantendo somente parte da rede base em que são utilizados os pesos) para treinar com uma base menor para um outro problema, como pode ser visto na sessão 2.2.

Outra forma de contornar o *overfitting*, resultante do uso de uma base pequena em uma rede grande, é através do uso de *data augmentation*, ou seja, aumentar o tamanho da base utilizando variados tipos de transformações nas imagens para criar mais imagens. As transformações mais tradicionais são: recorte de imagem (gerar imagens de regiões da imagem original), rotação, inversão de imagem (inverter horizontalmente e verticalmente).

2.1.2 Redes Importantes

AlexNet [33] é um dos trabalhos de CNN mais influentes em Computação Visual envolvendo redes neurais. É composta por 5 camadas de convolução e três totalmente conectadas. Após cada convolução de uma camada é usada a não linearidade Rectified Linear Unit (ReLU) como função de ativação de cada neurônio. A sua implementação só foi possível porque na época GPUs já estavam evoluindo para dar suporte a operações de convolução. Foram utilizadas duas GPUs para executar treinamento de metade de cada camada. Na Figura 2.5 pode-se ver a divisão de cada camada para cada GPU, sua distribuição de filtros e tipos de cada camada.

VGG [57] é inspirada na AlexNet e foi uma das primeiras arquiteturas a demonstrar o empilhamento de muitas camadas (16-19 camadas) de convolução com filtros pequenos (antes da publicação do artigo, era uma tendência aplicar filtros grandes como 11x11), como dois 3x3 seguidos resultam no mesmo campo receptivo de um filtro 5x5, produz melhor resultado por ter mais não-linearidade, pois são aplicadas em cada convolução, e ainda reduz a quantidade de parâmetros. A sua arquitetura de 16 camadas, conhecida

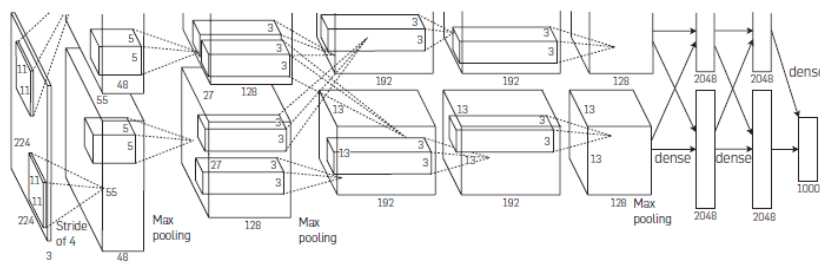


Figura 2.5: Arquitetura AlexNet (Fonte: [33])

como VGG16 é muito utilizada em diversos trabalhos, onde são aplicadas em um primeiro estágio para extração de características (na sessão 2.2 serão mostrados alguns exemplos).

Apesar de na teoria, a adição de mais camadas deveria resultar em melhor rede, na prática, a partir de certo ponto de profundidade, a acurácia da rede em seu treinamento não melhora, resultando em taxa de erro maior para redes mais profundas [25, 63].

Expandindo largamente os limites de profundidade de redes neurais, a **ResNet** [26] apresenta um módulo *Residual Block* que permite criar topologias que ultrapassam 100 camadas, sendo até menos complexas que a VGG, resolvendo o problema de degradação da acurácia em redes profundas. O módulo ResNet (Figura 2.6) consiste em um bloco de camadas que utiliza uma conexão *skip connection* que conecta o resultado da camada anterior ao bloco (representada por x na Figura 2.6) com a última do bloco na topologia da rede ($F(x)$), garantindo que pelo menos o que foi aprendido na camada anterior seja passado para as camadas mais à frente. Esta conexão é realizada com a adição da saída desta última camada do bloco (antes da aplicação de não-linearidade) com a saída da camada anterior ao bloco. Para a realização dessa adição, o tamanho de x e $F(x)$ devem ser iguais, portanto x é redimensionado quando necessário. Este módulo pode tanto ser utilizado em camadas convolucionais quanto em totalmente conectadas, e ao empilhar vários desses módulos, obtemos uma rede ResNet.

A adição de mais camadas a uma rede de convolução pode aumentar sua performance ao custo do aumento da quantidade de parâmetros, tornando a rede mais propensa a atingir *overfitting*, caso o conjunto de dados de treinamento não seja suficiente para seu tamanho. Além disso, o aumento de quantidade de camadas também exige mais recurso computacional para treinamento e inferência.

O módulo **Inception** [64], introduzido na arquitetura GoogLeNet, foi inspirado na ideia de compor camadas de convolução por redes neurais [36]. Ao invés de realizar uma convolução ou agrupamento em uma camada, ele une os dois, ou seja, a mesma entrada

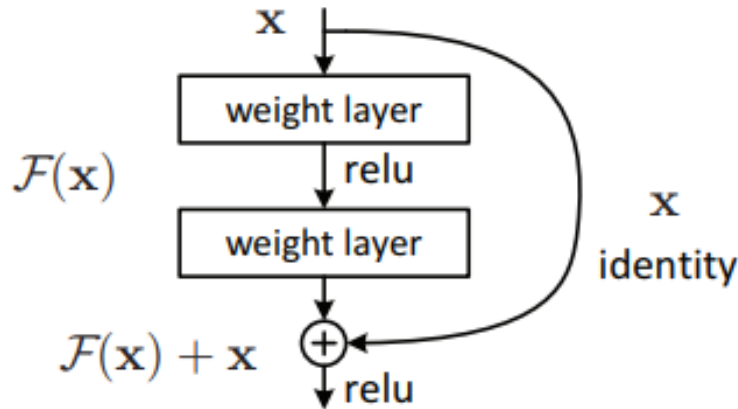


Figura 2.6: Módulo ResNet (Fonte: [26])

passa por convoluções 1x1, 3x3, 5x5 e por agrupamento, com o resultado de tudo concatenado em um lugar só (Figura 2.7). As convoluções 1x1, além de adicionarem mais não-linearidade (pois também aplicam RELU), possuem o papel principal de reduzir a profundidade dos volumes de mapas de ativação, garantindo o controle da quantidade de parâmetros, que a rede possa ter muitas camadas e que sua largura também possa ser aumentada sem muita penalidade em performance. Uma rede inception é uma rede composta por camadas de módulos inception, juntamente com algumas de *max pooling* com passo 2. A rede GoogLeNet possui 27 camadas, sendo que 22 delas possuem parâmetros treináveis.

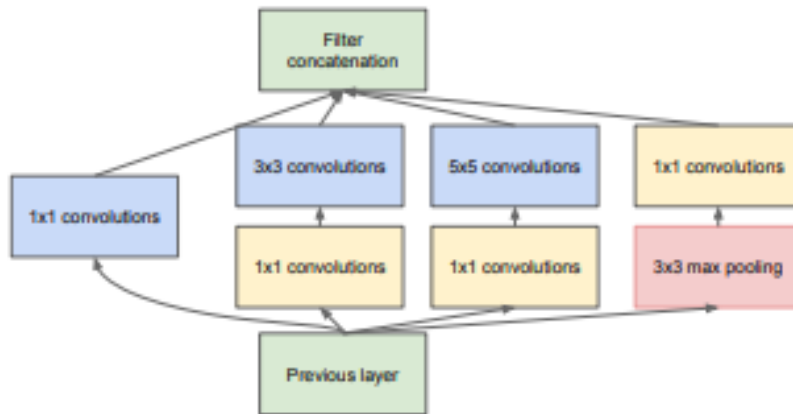


Figura 2.7: Módulo Inception (Fonte: [64])

Depthwise separable convolutions [56] são outra alternativa para diminuir a quantidade de parâmetros envolvidos nas operações de convolução. Onde ocorre primeiramente *depthwise convolutions* (Figura 2.8), que consiste em operações de convolução entre filtros para cada profundidade da entrada, ou seja, cada filtro é responsável por somente uma

profundidade e não todas, gerando assim um mapa de características para cada profundidade. Após isso, esses mapas de características são concatenados para passarem pela *pointwise convolution* (Figura 2.9), que consiste em convolução com filtro 1x1 com profundidade igual ao volume concatenado, fazendo com que as profundidades do volume sejam unidas, formando um mapa de características de profundidade 1. Esses dois passos reduzem drasticamente a quantidade de parâmetros e complexidade computacional do modelo.

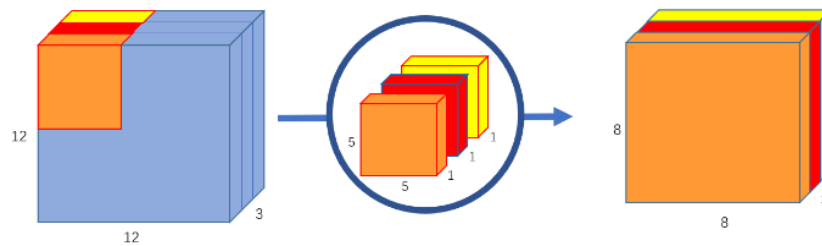


Figura 2.8: *Depthwise Convolutions* (Fonte: [71])

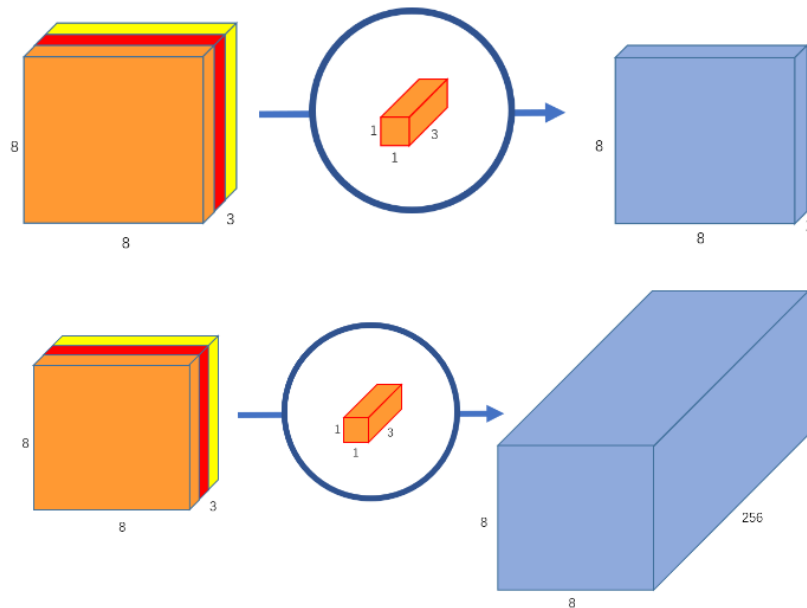


Figura 2.9: *Pointwise Convolution* (Fonte: [71])

Para entender como que a quantidade de parâmetros e operações são reduzidas com *depthwise separable convolutions*, deve-se calcular quantas operações ocorrem comparado com convoluções convencionais quando o objetivo é aumentar a quantidade de canais do mapa de características de saída da convolução. Visualizando a Figura 2.9, se o objetivo for ter como saída da convolução um volume 8x8 com 256 de profundidade, uma convolução convencional sem *padding* e passo 1, aplicada a uma entrada 12x12 com 3

canais seria composta por 256 filtros 5×5 de 3 canais cada. A cada passo da convolução de um filtro, seriam realizadas $5 \times 5 \times 3$ multiplicações, $5 \times 5 \times 3 \times 256$ (19.200 multiplicações) no total, após a aplicação dos 256 filtros. Como cada filtro vai se mover 8×8 vezes, o total de multiplicações consiste em $5 \times 5 \times 3 \times 256 \times 8 \times 8$ (1.228.800 multiplicações). Para extrair o mesmo volume com a *depthwise separable convolution*, primeiramente utilizamos 3 filtros $5 \times 5 \times 1$ para realizarem convolução na entrada, gerando um volume $8 \times 8 \times 3$. Esses filtros vão se mover 8×8 vezes, ou seja, $5 \times 5 \times 3 \times 8 \times 8$ multiplicações. A segunda etapa é a aplicação de 256 filtros $1 \times 1 \times 3$ na saída da etapa anterior (de tamanho $8 \times 8 \times 3$), cada filtro gerando uma saída $8 \times 8 \times 1$ que será concatenada com a saída dos outros filtros. Esta última operação possui $1 \times 1 \times 3 \times 256 \times 8 \times 8$ (cada filtro anda 8×8 vezes), ou seja, 49.152 multiplicações. Unindo a primeira etapa com a segunda, resulta em 53.952 multiplicações, bem menor que as 1.228.800 da convolução convencional.

A versão 3 de DeepLab (**DeepLabV3**) [10] utiliza *Atrous Spatial Pyramid Pooling* (ASPP) [9] juntamente com características em diferentes escalas utilizando convoluções dilatadas para transformar ResNet [26] em rede de segmentação semântica.

Como podemos observar na Figura 2.10, *output stride* representa a quantidade de vezes que o mapa de características é menor que a entrada e *rate* é o fator de dilatação das convoluções dilatadas. A parte ASPP (a) é composta por uma convolução 1×1 e três 3×3 convoluções dilatadas (todas com 256 filtros e *batch normalization*), nos quais os três fatores de dilatação são hiperparâmetros, ou seja, podem ser ajustados para treinamento e inferência. A operação *Image Pooling* é o resultado da aplicação de agrupamento com operação de média local, chamada *average pooling*, no mapa de características anterior juntamente com a aplicação de convolução 1×1 com 256 filtros e *batch normalization*, seguido de *upsampling* bilinear para a dimensão do mapa de características de entrada da ASPP.

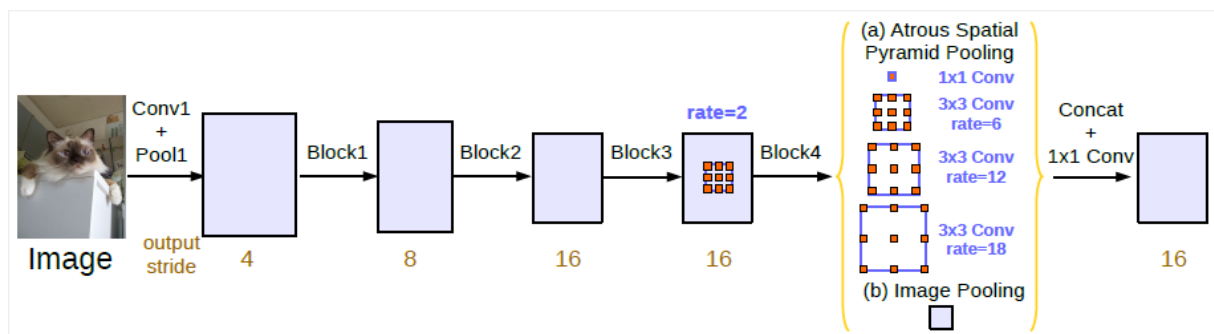


Figura 2.10: DeepLabV3 aplicada em ResNet (Fonte: [10])

Na DeepLabv3+ [7] a arquitetura de DeepLabv3 é alterada para seguir o modelo

Encoder-Decoder, onde a parte encoder consiste na DeepLabv3 com módulo ASPP sendo realizado com *atrous separable convolutions*. Essas *atrous separable convolutions* são a união de convoluções dilatadas com *depthwise separable convolution* (mencionada anteriormente). A saída da parte Encoder consiste no último mapa de características antes de *logits*. O módulo decoder, como visualizado na Figura 2.11, recebe o resultado do encoder com seu tamanho espacial aumentado com interpolação bilinear por uma razão de 4, que é concatenado com as características de baixo nível da parte encoder, que consiste no mapa de características anterior à aplicação de ASPP do encoder após a aplicação de convolução 1x1 para reduzir a sua quantidade de canais. Após a concatenação, algumas convoluções 3x3 são aplicadas para refinar o resultado, seguido de alteração do tamanho espacial final por interpolação bilinear, também com razão 4. Utilizando essa arquitetura, a melhor rede encoder em [7] consiste em uma versão modificada de Aligned Inception [13] (ou Aligned Xception).

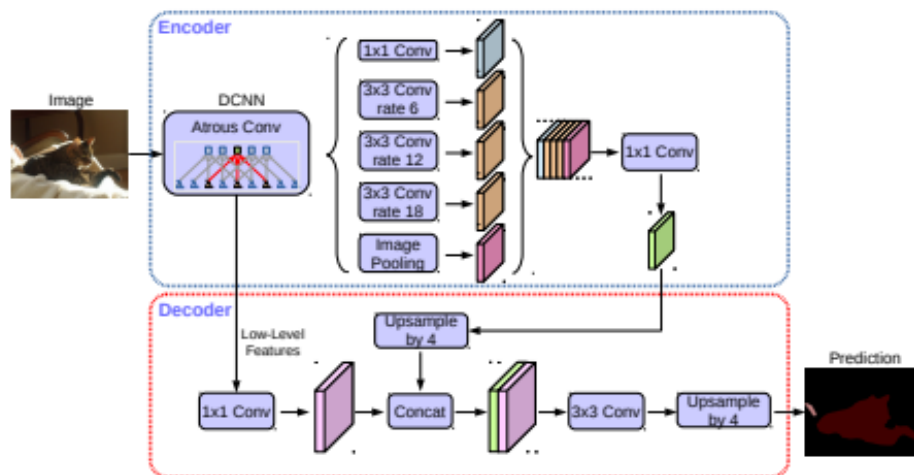


Figura 2.11: Arquitetura DeepLabv3+ (Fonte: [7])

A **Xception** original [12] utiliza módulos Inception modificados para terem *depthwise separable convolutions* no lugar das convoluções, onde não existe aplicação de não-linearidade entre as operações *depthwise* e *pointwise* (essa ausência gerou modelo que converge mais rápido e com mais performance). Além disso, utiliza as *skip connections* da ResNet. Sua arquitetura demonstrou ter 22,8 milhões de parâmetros, semelhante à Inception V3 [65] com melhor performance.

A versão Xception com a arquitetura DeepLabv3+ possui mais camadas que a Xception original [12], com mesma estrutura *entry flow* (módulo *entry flow* na Figura 2.12). Todas as operações de *max pooling* são substituídas por *depthwise separable convolution*. Também são adicionados *batch normalization* e ativações ReLU após cada 3x3 *depthwise*

convolution. Essa estrutura, sem a adição de ASPP e a parte decoder, pode ser vista na Figura 2.12.

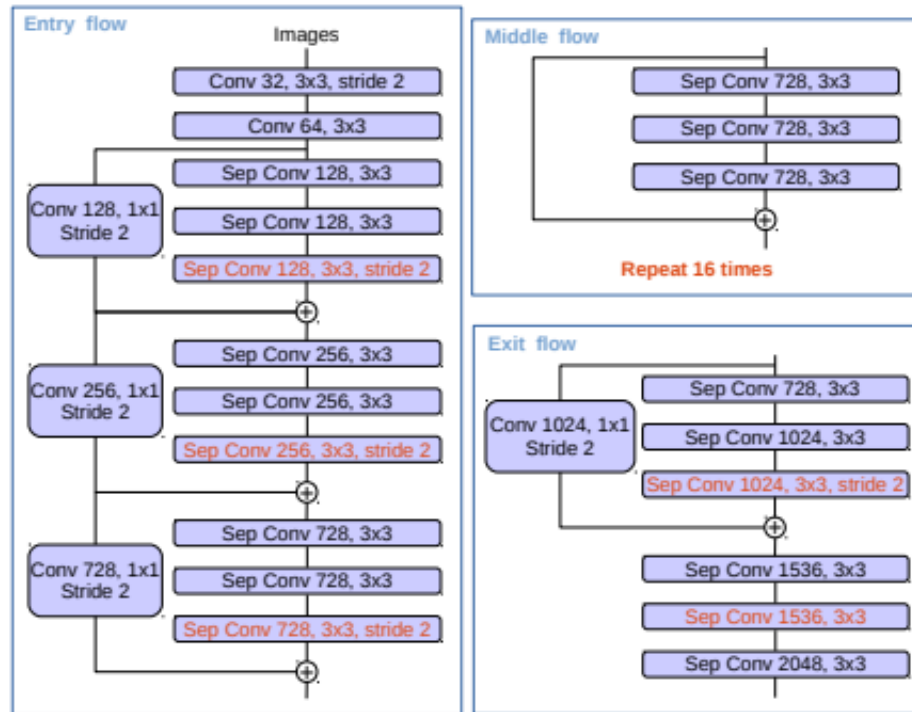


Figura 2.12: Arquitetura Xception Modificada (Fonte: [7])

MobileNet [27] também é composta por *depthwise separable convolutions*, além de possuir dois hiperparâmetros para ajustar seu tamanho: multiplicador de largura e de resolução. Esses multiplicadores podem ser usados para diminuir a largura da rede e resolução da entrada da rede, gerando versões menores do modelo MobileNet. Diferentemente de Xception, esta arquitetura não utiliza versões modificadas de módulos Inception e nem *skip connections* da ResNet, mas inclui a não-linearidade RELU após cada operação *depthwise* e *pointwise*. Sua quantidade de parâmetros também é bastante menor que a de Xception, 4,2 milhões, o que demonstra ser mais focada em sistemas computacionais com recursos de memória e processamento limitados.

Já **MobileNetV2** [54] é uma arquitetura derivada da sua primeira versão [27], também focada em diminuir a quantidade de parâmetros e operações, tornando possível sua utilização em ambientes com recursos de memória e processamento limitados, como dispositivos móveis. Seus componentes principais, e o que a torna mais eficiente do que sua primeira versão, são: *inverted residuals* e *linear bottlenecks*.

Ao contrário dos módulos residuais da ResNet [26], os módulos residuais da MobileNetV2 primeiramente aumentam a quantidade de canais que o módulo recebe como

entrada com convoluções 1×1 , para com o mapa de características resultante, realizar *depthwise convolution* diminuindo a quantidade de parâmetros, e finalmente realizar convolução 1×1 para modificar a quantidade de canais para a mesma da entrada. Esta operação é chamada de *inverted residuals* e pode ser visualizada na Figura 2.13(b), onde pode-se observar que a entrada e saída do módulo possui quantidade menor de canais do que as convoluções internas, sendo o contrário de um bloco residual normal (Figura 2.13(a)).

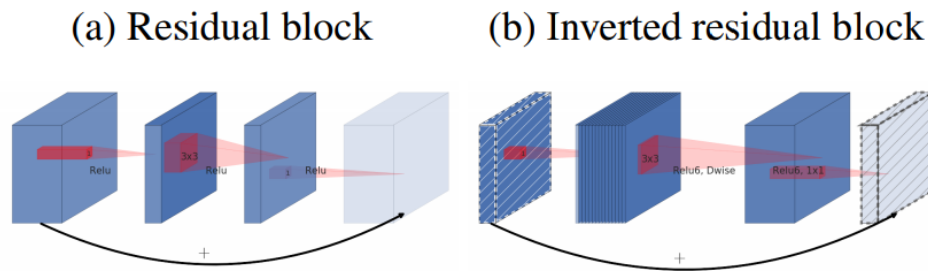


Figura 2.13: *Inverted Residuals* (Fonte: [54])

O *linear bottleneck* consiste em não realizar ativação não-linear antes da adição da saída do bloco com as ativações da entrada com a *skip connection*.

A Pyramid Scene Parsing Network (**PSPNet**) [76] tem como objetivo utilizar contexto global da imagem para realizar análise da imagem a nível de píxel. Sua estrutura é organizada em uma hierarquia de agrupamento que extraem mapas de características em diferentes escalas para serem concatenados e formar o mapa de características final.

A arquitetura geral pode ser visualizada na Figura 2.14, onde a imagem de entrada passa através de uma ResNet [26] pré-treinada com uma estratégia de *dilated network* [73, 8] para que o mapa de características seja extraído (Figure 2.14(b)). Ela então passa para o módulo de pirâmide de agrupamento (Figure 2.14(c)), composto por uma pirâmide de 4 níveis de filtros para a realização de agrupamento em diferentes escalas. O resultado de cada filtro tem sua dimensão em profundidade diminuída com a aplicação de convolução 1×1 e sua dimensão em altura e largura aumentada com aplicação de interpolação bilinear para que cada mapa de características tenha o mesmo tamanho do mapa de características de entrada da pirâmide. Esses mapas de características são então concatenados e mesclados com o mapa de características de entrada da pirâmide. A etapa final consiste em passar por uma camada de convolução, gerando o mapa de predição.

A Tabela 2.1 exhibe o resumo das redes descritas anteriormente, onde pode-se observar as principais características, quais são totalmente convolucionais (FCN), a quantidade de

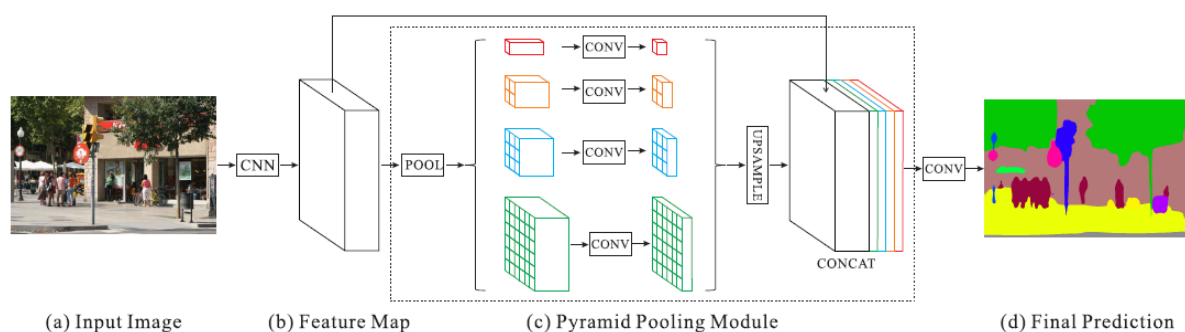


Figura 2.14: Arquitetura PSPNet (Fonte: [76])

camadas e o objetivo do uso de cada uma.

2.2 Segmentação de Imagens com Redes de Convolução

De acordo com [62], segmentação de imagem consiste em separar objetos ou regiões disjuntas de uma imagem, com a construção de um mapa espacial, onde há maior probabilidade de cada píxel pertencer a determinada classe. Nesta sessão falaremos como realizar segmentação de imagem com redes de convolução, sobre quais trabalhos são grande influência utilizando esta abordagem, além de descrever brevemente bases de imagens utilizadas nesse trabalho.

A área de segmentação de imagens é uma das mais estudadas no campo de Computação Visual. É um passo importante para a interação de robôs, ou aplicações computadorizadas, com o mundo real por meio de imagens. De acordo com [60], a segmentação pode ser realizada por uma das duas rotas básicas: métodos de arestas ou limites; métodos baseados em regiões. Os métodos de arestas detectam arestas entre regiões através da diferença entre grupos de píxeis. Já os baseados em regiões, atribuem píxeis para uma determinada região de acordo com o grau de similaridade.

A mais simples das técnicas consiste em usar um valor *limiar* T (Equação 2.1) para classificar cada píxel de uma imagem I verificando se a intensidade do píxel $I(x, y)$ atende a esse valor limite, se a imagem não for colorida; ou intensidade de cada canal de cor, caso seja colorida, sendo necessário um limiar para cada canal. A aplicação dessa técnica resulta em uma imagem binária, onde cada píxel $b(x, y)$ contém valor de intensidade 1, caso atenda ao limiar, ou 0, caso contrário. O limiar pode ser encontrado manualmente ou automaticamente, por meio de algum algoritmo que analise a imagem e o objetivo da segmentação para encontrar o melhor valor. Apesar de ser uma técnica simples e até

Tabela 2.1: Resumo comparativo das principais redes

Nome	Característica Principal	FCN?	Camadas	Objetivo
VGG [57]	Aumento de camadas com uso de filtros pequenos empilhados; VGG16, composta por 16 camadas, a mais famosa e utilizada	Não	16 a 19	Classificação
ResNet [26]	Introduziu a possibilidade de usar mais de 100 camadas com a aplicação do módulo ResNet	Não	101	Classificação
Inception [64]	Convoluções de várias escalas em uma mesma entrada, gerando uma saída unindo os resultados dessas convoluções; explorou uso de convoluções 1x1 para controlar quantidade de canais dos mapas de características	Não	27	Classificação
PSPNet [76]	Pirâmide de pooling em diferentes escalas e uso de ResNet transformada em FCN através da aplicação de convoluções dilatadas nas últimas camadas com output stride de 8	Sim	103	Segmentação Semântica
DeepLabV3 [10]	Uso de ResNet FCN através de convoluções dilatadas; Pirâmide ASPP de diferentes escalas de convoluções dilatadas + agrupamento aplicado à mesma entrada da pirâmide; output stride de 16	Sim	103	Segmentação Semântica
DeepLabV3+ [7]	Módulo encoder formado por uma rede para extração de características + ASPP composta de convoluções dilatadas + depthwise separable convolutions; Módulo decoder aplicado ao resultado do encoder; output stride de 16	Sim	Camadas do encoder + 2 do decoder	Segmentação Semântica
Xception [12]	Construída por módulos Inception compostos de depthwise separable convolution no lugar das convoluções do módulo Inception original, além de skip connections de módulos ResNet.	Não	36	Classificação
Xception + DeepLabV3+ [7]	Xception ocupa o lugar da rede de extração de características, com mais camadas (65 camadas) que a Xception original e operações de max pooling substituídas por depthwise separable convolutions; Módulo ASPP e decoder de DeepLabV3; output stride de 16	Sim	67 (65 da Xception + 2 do decoder)	Segmentação Semântica
MobileNet [27]	Composta por depthwise separable convolutions; possui variáveis para controlar a profundidade e largura da rede; focada em ser rápida e pequena; menor que a Xception	Não	28	Classificação
MobileNetV2 + DeepLabV3 [54]	Módulos ResNet compostos por depthwise separable convolutions, são o inverso de módulos ResNet convencionais; não realiza ativação não-linear antes da adição da saída do bloco com as ativações da entrada; expande últimos mapas de características como feito na DeepLabV3; não possui módulo ASPP e nem decoder; output stride de 16	Sim	19 (módulos bottleneck)	Segmentação Semântica

rápida de se executar, a variação de iluminação, a possível semelhança de cor do plano de fundo com o objeto da imagem e a não abrangência da relação espacial entre os píxeis são os pontos fracos dessa técnica e que podem atrapalhar seu resultado. Por esse motivo, é melhor aplicada quando quando plano de fundo e objeto de interesse possuem grande diferença de intensidade de píxel.

$$b(x, y) = \begin{cases} 1 & \text{se } I(x, y) > T \\ 0 & \text{senão} \end{cases} \quad (2.1)$$

Outra técnica bastante comum, é Crescimento de Regiões (do inglês *Region Growing*). Ela consiste em definir inicialmente um conjunto de píxeis aleatórios e adicionar píxeis iterativamente, que satisfaçam um critério de similaridade.

Para detectar arestas, muitas técnicas aplicam filtros específicos para diminuir ruído da imagem e deixar as arestas mais evidentes para um posterior algoritmo separar os objetos contidos nas arestas. Como falado no capítulo anterior, redes neurais de convolução são compostas por filtros. A vantagem do uso dessas redes é que elas aprendem a definir os valores desses filtros durante o treinamento. A cada iteração ela ajusta os valores dos filtros para que sua aplicação resulte em mapas de características que representem os objetos que ela visualiza nos exemplos do treinamento. Desta forma, a rede aprende filtros de arestas, texturas e até conceitos mais complexos, como falado antes.

Em [37], redes neurais de classificação profunda são adaptadas para seguir a arquitetura *Fully Convolutional Network* (FCN), onde todas as camadas da rede são convolucionais, além do uso de *skip connections* para a transmissão de informações de camadas mais iniciais para as últimas, fazendo possível ter uma saída com segmentação mais detalhada.

A idéia de que é possível transformar camadas totalmente conectadas em convolucionais vem do fato de que elas podem ser vistas como convolucionais com filtros que cobrem toda sua região de entrada. Na Figura 2.15 pode-se observar as últimas camadas totalmente conectadas de uma rede serem transformadas em convolucionais. A quantidade de nós de uma totalmente conectada se transforma na profundidade de uma camada convolucional, resultando em camadas $1 \times q$, sendo q a quantidade de nós da totalmente conectada. A saída dessa rede se torna em mapas espaciais de classificação. Por causa dessa característica, essa arquitetura pode ser usada para segmentação de imagens.

Um problema de converter redes totalmente conectadas em convolucionais é que, apesar de poderem receber como entrada imagens de qualquer tamanho, o da saída é reduzido por *subsampling*, que é usado por aquelas redes para diminuir a quantidade de processamento necessária para seu treinamento e inferência. Esse *subsampling* deteriora a qualidade da saída da rede convertida para convolucional, tendo como tamanho de saída relacionada com o tamanho de entrada em um fator igual ao passo utilizado nas unidades de saída. A forma que [37] resolve isso é por meio de *upsampling* com desconvo-

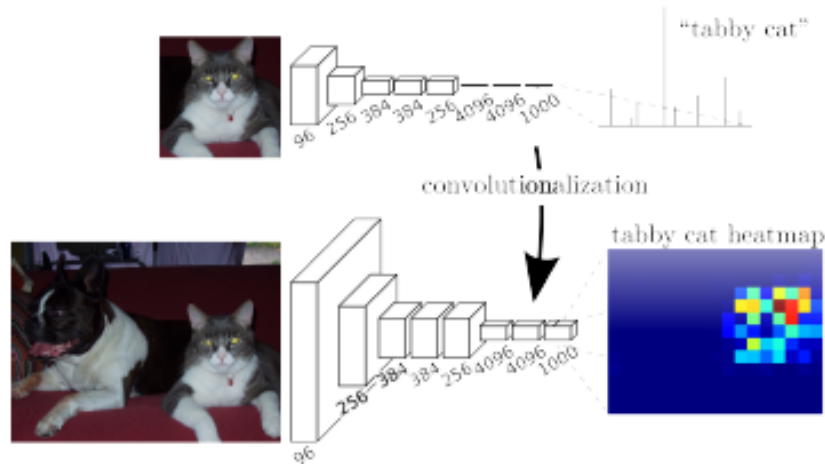


Figura 2.15: Arquitetura FCN (Fonte: [37])

lução, ou seja, aumentar as saídas, de camadas que passaram por convolução, utilizando desconvolução.

O processo de desconvolução consiste em operar convolução reversa nos passos *forward* e *backpropagation*, transformando o mapa de características resultante de uma operação de convolução no mapa de características de tamanho original antes da operação de convolução, que pode ser ajustado com o fator de passo de saída f . Este processo pode ser realizado por interpolação bi-linear ou ser aprendido durante treinamento (que inclusive foi citado em [37] como resultando em melhor desempenho).

A operação de realizar convolução com um filtro pode ser feita através da representação do filtro em forma de matriz de convolução (Figura 2.16.B), que consiste em uma matriz esparsa onde os elementos não-nulos são elementos do filtro posicionados de forma que a sua multiplicação (multiplicação de matrizes) com a representação vetorial (colocar cada elemento da matriz do mapa de características em um vetor de tamanho 16 no exemplo da figura) da matriz do mapa de características seja igual à convolução normal.

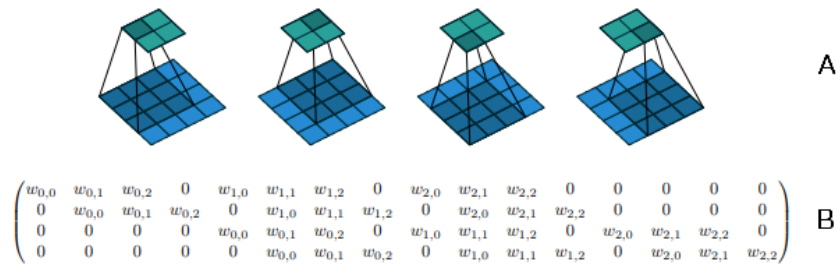


Figura 2.16: Matriz de Convolução (Fonte: [16])

A Figura 2.16 demonstra a construção da matriz de convolução de um filtro 3x3, com passo 1, sem *padding*. O mapa de características resultante da operação consiste em um

vetor de tamanho 4 que pode ser redimensionado para uma matriz 2x2 (Figura 2.16.A).

Organizando a convolução da forma acima, podemos realizar a desconvolução através da multiplicação da matriz transposta de convolução com o mapa de características resultante (redimensionado para vetor) da operação de convolução. Utilizando o exemplo acima, multiplicando a matriz de desconvolução 16x4 com um vetor de tamanho 4x1, obtemos um resultado de tamanho 16x1, que redimensionando para uma matriz 4x4, chegamos ao tamanho inicial do mapa de características antes da sua convolução.

Apesar da aplicação de *upsampling* com desconvolução conseguir aumentar a saída de camadas convolucionais, o resultado de segmentação ainda pode ser melhorado com a aplicação de conexões de *skip* [37], que consiste em combinar a camada de predição fina com camadas mais iniciais com passo menor, possibilitando a rede fazer predições locais respeitando a estrutura global das últimas camadas.

Para realizar *upsampling*, [44] aplica uma série de operações de desagrupamento e desconvolução. Para evitar que ocorra perda de informação espacial (que é necessária para realização de segmentação semântica) dos campos receptivos com a aplicação de agrupamento, [44] aplica desagrupamento [74, 75], que consiste em reconstruir o tamanho original dos mapas de características que passaram por agrupamento. Para que isso seja realizado, a localização e os valores das máximas ativações resultantes do *max pooling* são salvas em variáveis *switch* que são usadas para colocar essas ativações de volta nas posições corretas (Figura 2.17), reconstruindo as ativações para seu tamanho original de antes da aplicação de agrupamento. Desta forma desagrupamento reconstrói estruturas detalhadas de objetos nas imagens em melhores resoluções e camadas de desconvolução capturam detalhes específicos de cada classe, fazendo com que informações das imagens que pertencem às classes sejam realçadas.

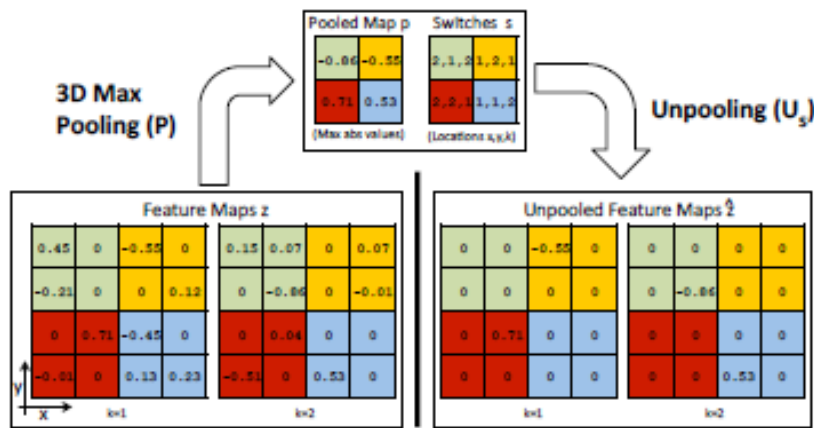


Figura 2.17: Agrupamento e Desagrupamento (Fonte: [75])

A maioria das redes de convolução para segmentação semântica eficientes atuais são baseadas de alguma forma na FCN [37]. O surgimento dessas outras redes baseadas em FCN forma um conjunto que pode ser classificado segundo características em comum: Encoder-Decoder, baseado no contexto e segmentação de instâncias. A seguir, falaremos de Encoder-Decoder e convolução baseada em contexto pois possuem mais semelhanças com as metodologias utilizadas nesse trabalho.

2.2.1 Encoder-Decoder

As arquiteturas em formato encoder-decoder possuem uma parte que produzem representações de baixa resolução ou mapas de características [22], chamadas *encoder*. A parte chamada *decoder* é responsável por decodificar o resultado da anterior para píxeis classificados nas classes de interesse do problema.

A **SegNet** [2] foi desenvolvida para interpretar cenas identificando ocorrências de objetos juntamente com contexto espacial para a realização de segmentação semântica. A arquitetura da parte encoder é topologicamente idêntica às 13 camadas convolucionais da rede VGG16 [58] (que são bastante comuns de serem utilizadas em arquiteturas encoder-decoder), porém, com as camadas totalmente conectadas removidas, o que torna a rede menor e mais fácil de treinar. Na Figura 2.18, sua arquitetura pode visualizada.

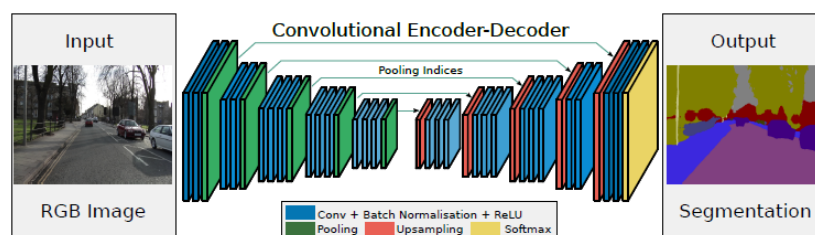


Figura 2.18: Encoder-Decoder da SegNet (Fonte: [2])

A parte decoder da rede é composta por uma hierarquia de decoders para cada encoder, distribuídos em 13 camadas. Cada decoder recebe os índices de *max pooling* correspondente ao seu encoder, para ser realizado o *upsampling* dos seus mapas de características de entrada. O uso desses índices no *upsampling* reduz a quantidade de parâmetros para serem treinados, tornando possível treinar toda a rede de início ao fim. A saída final dessa parte é usada como entrada para um classificador *softmax* de múltiplas classes para que retorne como saída uma imagem com quantidade de canais igual ao número de classes do problema, sendo que cada píxel contém a probabilidade máxima de pertencer à determinada classe. Sua arquitetura pode também ser facilmente aplicada em qualquer rede com

arquitetura encoder-decoder.

A principal diferença entre o *upsampling* da SegNet e o da FCN está no fato de que a SegNet não treina o *upsampling* que realiza usando índices de *max pooling*, somente depois de *upsampling* que aplica um filtro de vários canais treinável. A FCN realiza *upsampling* através de deconvolução por meio de interpolação bilinear. FCN também realizou *upsampling* por meio de características que são treinadas para realizar deconvolução. Além disso, ela armazena os mapas de características da parte encoder durante inferência, o que torna necessário o uso de muita memória para armazená-los.

Outro exemplo dessa arquitetura é [44], que primeiramente detecta propostas de objetos na imagem com [77] para aplicar a rede nestas imagens dos objetos, depois unir os resultados no tamanho da imagem original.

2.2.2 Baseada no contexto

Outra forma de orientar a segmentação é utilizar o contexto global da imagem para resolver ambiguidades locais. O contexto global é perdido na aplicação de agrupamento, apesar de ele conseguir deixar a rede indiferente em transformações espaciais dos mapas de características. Mantendo a arquitetura convolucional, podemos conseguir contexto global com *Conditional Random Fields* (CRFs), convoluções dilatadas e *multi-scale aggregation*.

Ao utilizar **CRF** como pós-processamento para suavizar ruídos de mapas de segmentação, pode-se observar melhores resultados na segmentação. [8] utiliza um CRF totalmente conectado [34], conseguindo 4% de melhora na validação, fazendo com que o modelo capture mais detalhes de contorno de objetos comparado com o modelo sem a aplicação de CRF. Algo semelhante é encontrado em [3].

Quanto maior o campo receptivo das camadas, maior é a informação de contexto que a rede é capaz de capturar. **Convolução Dilatada** (do inglês *Dilated Convolution*) [73], também chamada *Atrous Convolution* é outra alternativa para expandir o campo receptivo, que é realizada através do aumento da taxa de dilatação. Esse processo pode ser feito através da expansão do tamanho dos filtros de acordo com essa taxa e acrescentando zeros entre os valores originais do filtro. Na Figura 2.19 pode-se visualizar que com taxa de dilatação 1 ($D=1$), temos o filtro original, com valores não-nulos em azul, e a medida que se aumenta esse fator, mais zeros (campos em branco) são acrescentados entre os elementos originais. Depois da expansão dos filtros, basta aplicar convolução usando-os nos mapas de características para que o processo seja concluído.

Na Figura 2.20 os pontos vermelhos são os elementos não-nulos dos filtros e os campos em verde a representação do campo receptivo relacionado com o filtro em questão. Na medida que se aumenta a taxa de dilatação e empilha um filtro com outro, maior é o tamanho do campo receptivo do filtro mais acima. Se empilharmos o filtro (a) com o filtro (b), observa-se na Figura 2.20 que o campo receptivo em (b) é de tamanho 7x7 (parte verde na imagem), ou seja, tem um alcance maior na imagem de entrada, conseguindo extrair mais informação de contexto. Esse campo receptivo aumenta mais ainda se empilharmos logo após (b) o filtro (c), alcançamos um campo receptivo de 15x15 em (c). Ou seja, com a aplicação de convoluções dilatadas, podemos atingir campos receptivos maiores sem precisar empilhar muitas camadas de convolução.

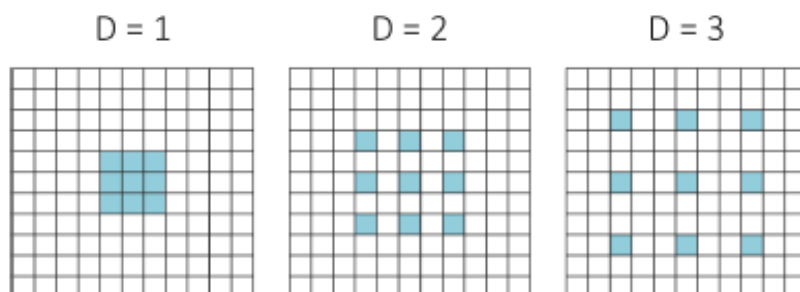


Figura 2.19: Dilatação de filtros (Fonte: [22])

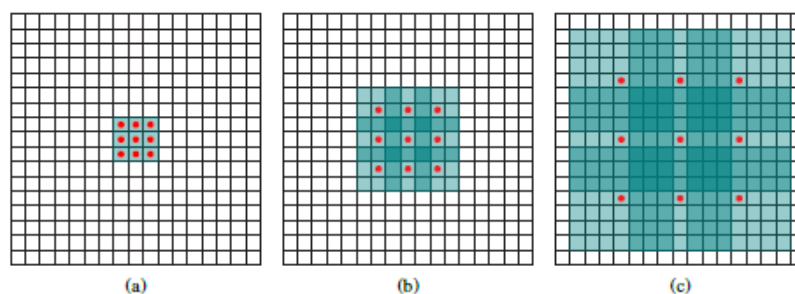


Figura 2.20: Expansão de Campos Receptivos relacionados com os filtros (Fonte: [73])

Outra forma de capturar mais o contexto da imagem a ser segmentada é por meio do uso de predição em múltiplas escalas, que consiste em utilizar **redes de múltiplas escalas**, nas quais são compostas por redes que atuam em escalas diferentes para que o resultado seja unido no final da inferência.

[51] realiza segmentação em múltiplas escalas através do uso de duas redes de forma semelhante à [17]. Uma é responsável por prever um resultado global bruto. Esta consiste em uma VGG16 com suas camadas completamente conectadas substituídas por uma camada de convolução constituídas de pequenas redes neurais, chamadas Network-In-

Networks [36], no lugar de filtros lineares, o que permite extrair características melhores por ser não-linear. A outra é uma modificação da primeira, responsável por receber o resultado bruto da primeira e refinar o resultado. A diferença em escalas está no tamanho do campo receptivo da primeira, que consiste no tamanho total da imagem de entrada, ou seja, abrange um contexto maior; a segunda possui campo receptivo menor.

2.2.3 Bases de Imagens

Existem diversas bases de imagens para segmentação de imagens que são parte de desafios em diferentes tipos de problema em segmentação. Nesses desafios, pesquisadores submetem suas aplicações de segmentação à avaliação definida nessas bases de dados. A seguir, falaremos das bases públicas que foram utilizadas nesse trabalho: PASCAL VOC [20, 18], FPVRGame [46], GTEA [21], SFA [6], ViolaSet [68].

2.2.3.1 Pattern Analysis Statistical Modeling and Computational Learning Visual Object Classes (PASCAL VOC)

Entre 2005 e 2012, esta base de dados recebia novos conjuntos de imagens e anotações, representando o desafio do ano, onde pesquisadores submetiam seus métodos para avaliação automática. O mais recente (PASCAL VOC 2012) conjunto de imagens possui 20 classes (excluindo plano de fundo), 11530 imagens com 27450 anotações de objetos em formato ROI e 6929 segmentações.

Os desafios são classificados em três principais grupos: classificação (identificar se existe determinado objeto em uma imagem), detecção (detectar a localização das instâncias de um objeto de uma classe) e segmentação (identificar as classes que cada píxel da imagem pertence). Além de outros sub-grupos como: classificação de ações (identificar quais ações são praticadas por uma pessoa), contorno de pessoa (identificar onde estão cabeça, mãos e pés).

As suas 20 classes são distribuídas em 5 subgrupos:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

2.2.3.2 FPVRGame

Base de imagens de braços e mãos em primeira pessoa, composta por 58.868 amostras com 320x240 de tamanho, retiradas de vídeos, capturadas em ambiente interno, com 15 indivíduos (11 homens e 4 mulheres), incluindo imagens espelhadas. Cada indivíduo possui imagens com 7 gestos realizados com os braços e mãos (Figura 2.21).

Utilizamos conjuntos de imagens de cada indivíduo para compor a base Egohand (4041 imagens), incluindo todos os gestos, mas excluindo as imagens espelhadas.

Class	Game action	Captured RGB images
0	background	
1	move left	
2	move right	
3	move forward	
4	move back	
5	select	
6	pick up	
7	jump	

Figura 2.21: Gestos da base FPVRGame (Fonte: [46])

2.2.3.3 Georgia Tech Egocentric Activity Datasets (GTEA)

Seu foco está em aplicações de detecção de gestos em atividades rotineiras em perspectiva de primeira pessoa. Com o passar dos anos, a base foi aumentando seu conjunto de imagens, gerando 4 conjuntos: GTEA, GTEA Gaze, GTEA Gaze+, EGTEA Gaze+.

O conjunto EGTEA Gaze+, o maior grupo e mais atual, é composto por 28 horas de atividades relacionadas a cozinhar de 86 sessões únicas de 32 sujeitos diferentes; áudios e informações de direção do olhar de sujeitos; anotações das ações dos sujeitos e 15176 máscaras de mãos de 13847 quadros de vídeos.

2.2.3.4 SFA

É uma base de imagens de pele humana baseada em duas clássicas bases de rostos: FERET [48] e AR [40]. Ela contém imagens de rosto de pessoas com diferentes tons de cor de

pele, idade e sexo.

As imagens são divididas em quatro grupos: imagens originais (1118 imagens), máscaras de pele (1118), amostras de pele (3354), amostras de não-pele (5590). As amostras são imagens em diferentes tamanhos com somente cores de pele ou não-pele extraídas das 1118 imagens originais. As máscaras (mesma quantidade de imagens originais) representam regiões de pele nas imagens originais.

2.2.3.5 ViolaSet

Ela é composta por mais de 3000 imagens de pessoas (corpo e rosto) em diferentes ambientes e tamanhos, com plano de fundo não controlado para segmentação de pele humana. Cada imagem possui máscara para classificar cada píxel em pele ou plano de fundo. Utilizamos somente 2922 pois removemos as em formato de GIF ou corrompidas.

Capítulo 3

Trabalhos Relacionados

Visão em Primeira Pessoa (do inglês FPV - First Person Vision) possui múltiplos desafios [4] como: (1) câmeras não-estáticas, (2) iluminação e plano de fundo variável, (3) processamento em tempo real (4) capacidade de processamento de vídeo reduzida. Para que seja realizada segmentação de mãos e braços em contexto FPV, a técnica de segmentação deve contornar esses desafios. Nesta sessão, mostraremos trabalhos que estudam formas de segmentar imagens de braços ou mãos em visão egocêntrica.

Aplicando técnicas de Aprendizado de Máquina mais tradicionais em Visão Computacional, [42] (Figura 3.1) trata esse problema de segmentação de mãos em visão egocêntrica com o desenvolvimento de um método executado em uma aplicação de Google cardboard/HMD utilizando uma combinação de equalização de histograma, Gaussian blur e seu proposto Multi Orientation Matched Filtering (MOMF). Porém, apesar de atingir inferência em tempo real, a acurácia de segmentação é 6,97% pior que SegNet [2]. Já [35] (Figura 3.2) combina modelos de aparência global com uma combinação esparsa de dimensão 50 de cores, textura e características de histograma de gradiente para segmentar mãos de imagens sobre iluminação variada e diferentes poses de mãos.

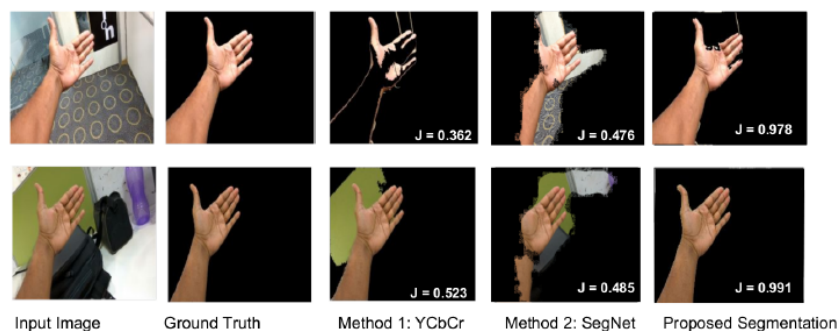


Figura 3.1: Exemplo de Maurya et al. [42]. (Fonte: [42])

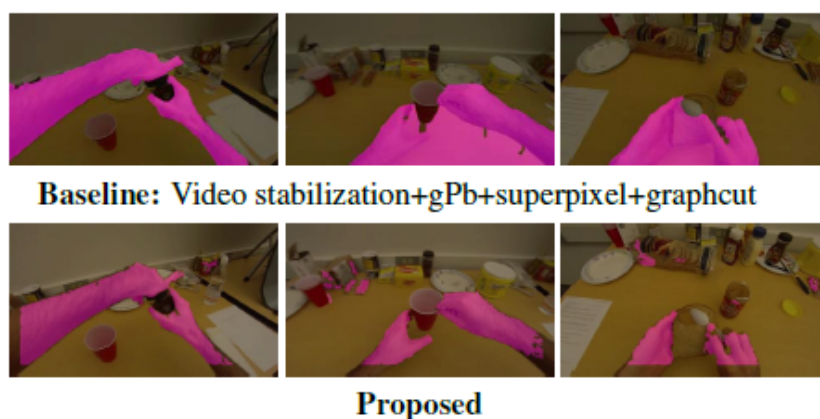


Figura 3.2: Exemplo de Li et al. [35]. (Fonte: [35])

No campo de estudo de redes neurais profundas, [55] (Figura 3.3) aplica a rede neural de convolução YOLO [52] para detectar mãos em cenários de condução de carros para reconhecer o que o punho do motorista faz durante a condução. Apesar de seu treinamento ter sido realizado com uma quantidade grande de imagens de mãos, a base de imagens não tinha como foco somente visão em primeira pessoa.



Figura 3.3: Visão geral de Siddharth et al. [55]: (a) Imagem original (b) Detecção de mãos (c) Classificação de pele a nível de píxel (d) Extração de mão localizada (e) Classificação de intenção do motorista. (Fonte: [55])

Auto-Context [66] consiste em um método de segmentação iterativa, em que o resultado de cada iteração é utilizado na próxima iteração juntamente com a imagem de entrada original. Outra abordagem [70] (Figura 3.4) desenvolve uma rede neural de convolução focada em segmentação de mãos em imagens na perspectiva de primeira pessoa em tempo real, diferenciando de Auto-Context por segmentar inicialmente em uma versão da imagem original de escala menor com o resultado passado para a próxima iteração após realização de *upscaling*. Neste trabalho foi criada uma base de imagem similiar (porém menor) a nossa pelo fato de obterem como cenário ambientes externos e um interno de escritório, com 348 imagens, sendo que 90% delas foram utilizadas em treinamento e o resto para testes.

DeepLabV3 [10] e PSPNet [76] não foram desenvolvidas inicialmente para serem usadas em ambiente de visão em primeira pessoa. Porém, a PSPNet está atualmente em

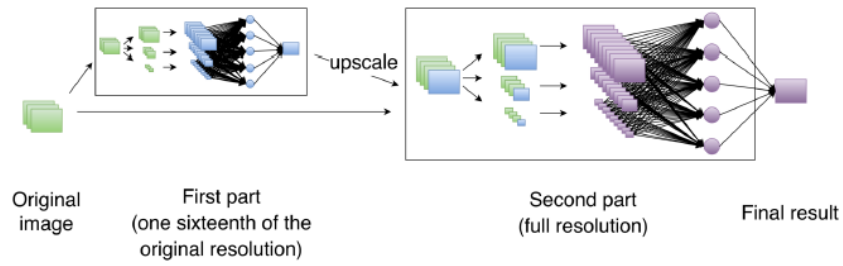


Figura 3.4: Visão geral da arquitetura de Vodopivec et al.. (Fonte: [70])

terceiro lugar na base de dados PASCAL VOC 2012; e a DeepLabV3+ [7] ainda em primeiro, representando o estado da arte atual. Neste trabalho, essas arquiteturas são implementadas para segmentar imagens de mãos e braços humanos em visão egocêntrica.

Outra forma de obter segmentação é através do uso de câmeras de profundidade que capturam informações de profundidade dos objetos na cena. Essas informações podem ser combinadas com as cores dos pixels de imagens de câmeras convencionais para ajudar algoritmos classificadores de pixels a separar melhor os objetos alvos [24, 50, 37, 61].

A Tabela 3.1 exibe um resumo comparativo dos principais trabalhos relacionados mencionados acima, onde pode-se observar as características e os pontos fracos de cada um, o que motiva nosso trabalho em desenvolver uma técnica de segmentação que não tenha esses pontos fracos.

Tabela 3.1: Resumo comparativo de trabalhos relacionados

Nome	Descrição	Ambiente Aplicado	Ponto Fraco	Velocidade em ms
Maurya et al. [42]	Filtro MOMF com Limiar de cor	<i>Google Cardboard</i> em <i>Smartphones</i>	Qualidade de segmentação 6,97% pior que SegNet e uso de intervalo de cor limita muito a quantidade de cores de pele	36
Li et al. [35]	Combina características locais com modelagem de aparência global.	PC	Falta melhor descrição do conjunto de imagens públicas que foi utilizado.	Não informado
Siddharth et al. [55]	Rede Neural Profunda YOLO	PC	Não segmenta em somente em imagens egocêntricas; Imagens somente sobre interior de carros. Objetivo principal não é segmentação, mas classificação de gestos.	27,7
Votopivec [70]	Rede Neural Profunda de arquitetura própria com penúltima camada totalmente conectada	PC	Imagens de treinamento capturadas por câmera segurada pelas mãos do usuário, diferenciando do posicionamento correto na região dos olhos; Quantidade de imagem de treinamento pequena, podendo causar overfitting.	39,2
PSPNet [76]	Uso de pirâmide de pooling para aprimorar resultado obtido da ResNet totalmente convolucional	PC	Não focada em segmentação de braços e mãos em primeira pessoa	Não informado
DeepLab [10]	Uso de pirâmide de convoluções dilatadas unidas com <i>depthwise separable convolutions</i> com posterior aplicação de decoder	<i>Mobile</i>	Não focada em segmentação de braços e mãos em primeira pessoa	Não informado

Capítulo 4

Segmentação de Braços Humanos em Visão Egocêntrica com PSPNet E DeepLab

Em segmentação semântica, PSPNet [76] e DeeplabV3 [10] alcançaram o estado da arte na base de imagens PASCAL VOC Challenge 2012 [19], e dentre diversas categorias de imagens, essa base contém imagens de pessoas em diversos ambientes. Por este motivo, as técnicas de segmentação de imagem que obtiveram estado da arte nesta base podem ser classificadores eficientes para pixels que representam pessoas ou partes delas.

PSPNet e DeeplabV3 também apresentaram conceitos que ajudaram a evolução de segmentação semântica utilizando redes neurais profundas. Por causa dessas conquistas, focamos em realizar *fine-tuning* nessas arquiteturas para aplicá-las em classificação binária de pixels de braços humanos e plano de fundo, ou seja, separar pixels de imagem que representam um braço humano dos pixels que representam o plano de fundo. A seguir, apresentamos rapidamente as arquiteturas e também mostramos como realizamos o *fine-tuning* para atingir nosso objetivo particular.

4.1 PSPNet

Em nossa implementação dessa arquitetura, a camada "conv6" (nome da camada no arquivo "prototxt" de definição da rede para treinamento no *framework* Caffe [30]) e todas as camadas seguintes foram treinadas com saída de tamanho 2, que representa a quantidade de classes do problema aqui proposto: braço ou não-braço. A ResNet101 [26] (ResNet com 101 camadas) foi utilizada para extrair o mapa de características de entrada da pirâmide de agrupamento, sendo que [76] modifica suas últimas camadas para serem convoluções dilatadas possibilitando o *upsampling* dos mapas de características finais, pois a ResNet

é originalmente focada em classificação e não segmentação semântica.

4.2 Deeplab

A implementação de Xception utilizando a arquitetura DeepLabv3+ utilizada neste trabalho, composta por módulos ASSP e decoder, se chama Xception X-65 [7] (por ser a configuração composta por 65 camadas).

A implementação da MobileNetV2 [54] com a arquitetura DeepLabV3 foi utilizada neste trabalho. Sendo que no trabalho MobileNetV2 [54], foi detectado que a rede reduz grandemente a quantidade de operações realizadas ao retirar o módulo ASPP, por esse motivo, também utilizamos a implementação sem este módulo. Essa implementação também não possui *decoder* da arquitetura DeepLabv3+.

Para esses dois encoders, Xception X-65 e MobileNetV2, os pesos da camada de classificação não são re-utilizados e a última camada da rede somente contém *logits* (um parâmetro da implementação original em que pode-se ativar ou não, e que vimos que ativado resulta em melhor resultado). Além disso, o modelo treinado é exportado para trabalhar com somente duas classes ao invés das 21 padrão na rede original.

4.3 Configurações dos Treinamentos

Para cada configuração de rede, foram realizados cinco treinamentos para capturar as métricas de avaliação. As configurações da PSPNet foram treinadas por 100 mil iterações (50 epoch de 2000 passos de gradiente) e cada configuração da DeepLab por 30 mil. Esses valores representam a configuração *default* de treinamento de cada arquitetura nos seus artigos originais, que não foram modificados para verificar os resultados sem realizar muito esforço em modificar iterações. Cada uma dessas configurações se diferencia em hiperparâmetros e banco de imagem utilizado.

4.3.1 Bancos de Imagens

Quatro bancos de imagens foram utilizados. Na Figura 4.1 podemos observar alguns exemplos de imagens de cada um. Dois deles (ViolaSet [68] e SFA [6]) foram mesclados para criar outro (Huskin) para representar uma base de imagens em variadas perspectivas.

O conjunto de imagens EgohandVal foi criado a partir de imagens que capturamos em

perspectiva de primeira pessoa posicionando um celular *Samsung Galaxy J7 Plus* em um dispositivo de realidade virtual que segue a arquitetura *Google Cardboard*, posicionado na cabeça do usuário, sendo que a câmera do celular fica exposta ao que a pessoa está vendo à sua frente. Com esse dispositivo, foi possível capturar as imagens na altura e posição dos olhos do usuário. A seguir, são exibidas as informações gerais da composição das bases que utilizamos:

- ViolaSet [68] (Figura 4.1 (A)). Foram utilizadas 2922 imagens resultantes da remoção de GIFs e imagens corrompidas.
- SFA [6] (Figura 4.1 (B)). Foram utilizadas todas as imagens originais com suas correspondentes máscaras de pele.
- Egohand (Figura 4.1 (C)). Possui 4041 imagens extraídas da base FPVRGame [46], na qual criamos as máscaras para cada imagem com a aplicação de segmentação com Grabcut [53]. As imagens possuem tamanho 320x240 e representam pessoas em perspectiva egocêntrica realizando um conjunto de gestos em ambientes internos.
- EgohandVal (Figura 4.1 (D)). Esta é uma base totalmente criada neste trabalho, utilizada para validação da performance dos treinamentos nas anteriores. É composta por 322 imagens com 640x360 de tamanho, capturadas de 4 indivíduos: 2 homens de cor de pele branca em ambientes internos; 2 mulheres de cor de pele negra e 1 de cor de pele parda, todas em ambiente externo. Esses indivíduos e ambientes não estão presentes na base anterior, porém os gestos que realizam são semelhantes. Suas máscaras foram também criadas com a aplicação de Grabcut.

Mesclando ViolaSet e SFA, compomos o conjunto de imagens que chamamos Huskin (imagens que contêm pele humana), com 4040 imagens no total. Esse conjunto foi utilizado para verificar se treinando com imagens de pessoas em diferentes perspectivas gera modelos para segmentação em perspectiva egocêntrica melhores que os treinados com imagens somente na perspectiva egocêntrica (base Egohand). *Data augmentation* é também aplicado em cada conjunto de treinamento pelo motivo de as arquiteturas utilizadas terem muitos parâmetros para a quantidade de imagens.

Cada configuração de rede foi treinada com a base Huskin e com a base Egohand, gerando assim dois experimentos separados. Os modelos resultantes do treinamento são validados com a base EgohandVal. Assim, verificamos se o treinamento com Huskin é melhor que o feito com a Egohand.

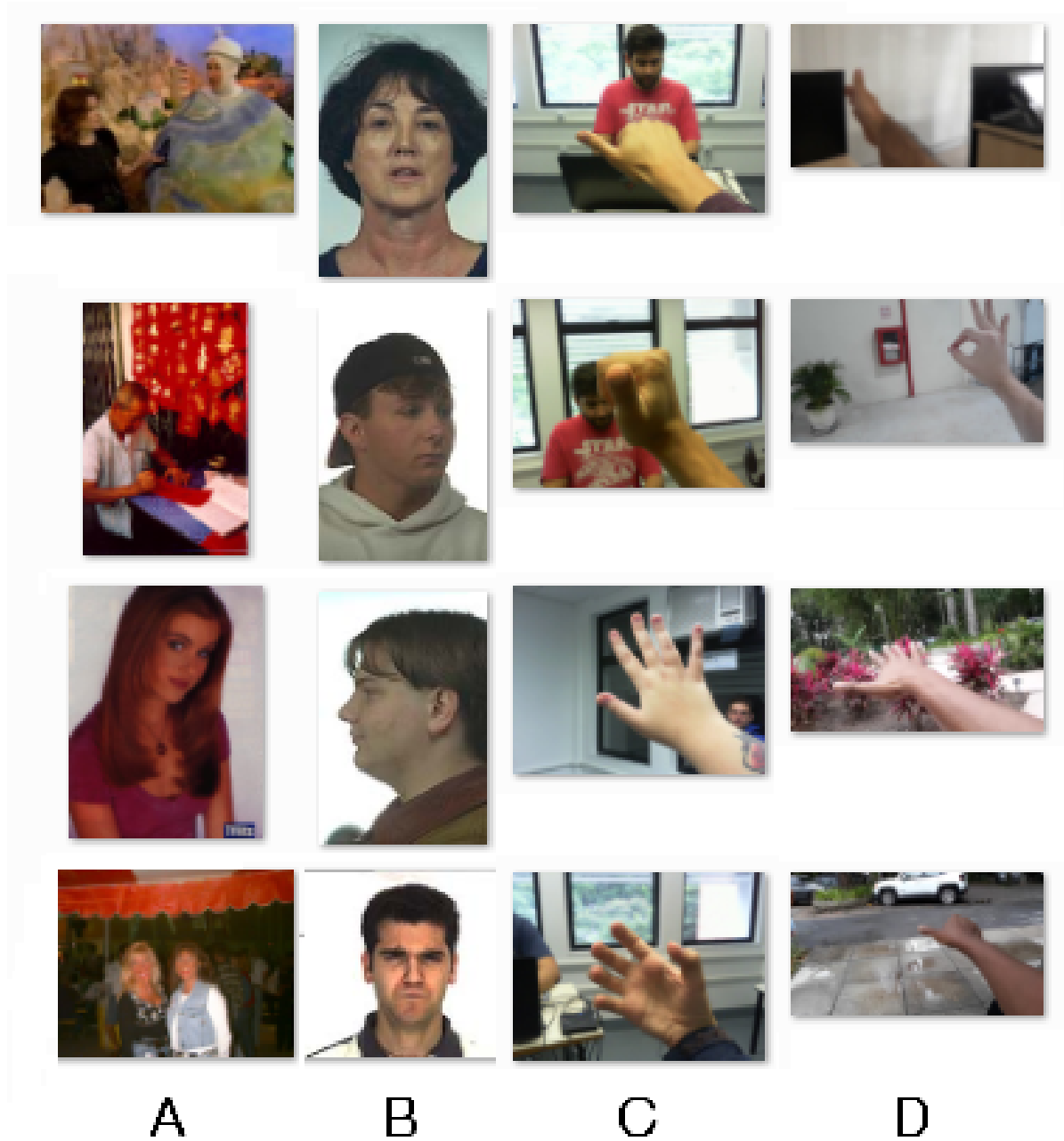


Figura 4.1: Exemplos de cada banco de imagens. (A) ViolaSet, (B) SFA, (C) EgoHand, (D) EgoHandVal.

4.3.2 Configurações de treinamento de DeepLab

Foram utilizados pesos pré-treinados e publicamente disponíveis de DeepLab na base PASCAL VOC 2012 [19] para inicializar os pesos do treinamento da nossa implementação de DeepLabv3+ com Xception como encoder e DeepLabv3 com MobileNetV2.

Durante o treinamento, a taxa de aprendizado obedeceu o mesmo padrão utilizado no trabalho [10], que consiste em seguir a política "poly", onde a taxa de aprendizado inicial

de 0,0001 é multiplicada por $\left(1 - \frac{iter}{max_iter}\right)^{power}$ com $power = 0,9$, $iter$ representando a iteração atual e max_iter a iteração máxima e $weight\ decay$ 0,00004.

As imagens e máscaras são redimensionadas para 513x513 durante treinamento e validação.

Os blocos da arquitetura Deeplab adicionados às arquiteturas bases são todos treinados com *batch normalization*, *output stride* de valor 16 em todas as configurações de redes Deeplab implementadas nesse trabalho. Quando Xception é usado, o *output stride* do decoder é definido como 4 e as taxas de dilatação do módulo ASPP são definidas como 6, 12 e 18 (tanto em treinamento quanto validação). Durante o treinamento variamos o tamanho do *batch* para os dois encoders, deixando o resto dos hiperparâmetros como definidos acima, sendo que os não mencionados são deixados com seus valores padrão:

MobileNetv2 - *depth multiplier* de 1; realizando *fine-tuning* de *batch normalization*; *output stride* foi definido como 16 por ter sido o que resultou melhor resultado em [54].

Xception - também deixamos *output stride* da parte encoder como 16 por ter sido o valor que obteve melhores resultados em processamento e acurácia em [7]; *fine-tuning* de *batch normalization* também é realizado com *decay* de 0,9997, *epsilon* (variável utilizada para impedir divisão por zero ao normalizar ativações com suas variâncias) de 1e-3 utilizando multiplicador *gamma* para escalar as ativações no *batch normalization*.

O *data augmentation* utilizado consiste em modificar a escala das imagens aleatoriamente entre 0,5 a 2,0, além de inversão horizontal e vertical também aleatória.

Como a implementação original de Deeplab [10] foi feita em Tensorflow [1], também utilizamos esse *framework* em nossa implementação.

4.3.3 Configurações de treinamento PSPNet

O *fine-tuning* foi também realizado com pesos pré-treinados com PASCAL VOC 2012 [19]. A taxa de aprendizado foi definida como fixa em 1e-10, *momentum* 0,99, tamanho de *batch* de uma imagem e *weight decay* de 0,0005.

As imagens e suas máscaras são redimensionadas para 473x473 durante treinamento e validação, que consiste no tamanho da entrada da rede.

O *data augmentation* foi configurado para ser semelhante ao realizado em DeeplabV3. Portanto, as operações de *data augmentation* da implementação TensorFlow da DeeplabV3 foram reproduzidas na linguagem de programação Python, para que fosse possível

utilizá-las na implementação da PSPNet em Caffe [30].

Capítulo 5

Resultados e Discussão

Neste capítulo serão exibidos os resultados dos experimentos realizados, que consistem no teste de inferência dos modelos treinados utilizando as diferentes configurações das arquiteturas DeepLab e PSPNet, citadas na sessão 4.3, no conjunto de imagens de validação composto por imagens em perspectiva egocêntrica (EgohandVal). O desempenho dos experimentos são medidos através do uso das métricas: acurácia média, F-Score (média harmônica da precisão e taxa de *recall*) [11], Matthews Correlation Coefficient (MCC) [41], e seus respectivos desvios padrão resultantes dos cinco treinamentos para cada modelo treinado.

Focamos a comparação da performance de segmentação com MCC, apesar de F-Score e acurácia terem sido capturados para que possam ser usados em outros estudos. Os melhores modelos nas duas arquiteturas, de acordo com a métrica MCC, serão utilizados para testar o desempenho na base de imagens pública GTEA.

O que esperamos dos experimentos é exibir que os modelos treinados utilizando a perspectiva egocêntrica possuem melhores resultados de inferência em imagens egocêntricas do que aqueles treinados com imagens em diferentes perspectivas, além de uma velocidade de segmentação que seja possível sua execução em tempo real. Ao final do capítulo serão exibidas as imagens resultantes da segmentação do conjunto de validação e do conjunto GTEA utilizando os melhores modelos.

Pela natureza desbalanceada da quantidade de pixels que pertencem ao plano de fundo e que pertencem a um braço humano em uma imagem, simplesmente capturar a acurácia de segmentação pode esconder a real performance do modelo utilizado, pois braços humanos não costumam tomar grande parte de uma imagem capturada na perspectiva egocêntrica. Por isso, o modelo pode estar acertando muitos pixels de plano de fundo e

poucos de braços. Este problema é muito comum no domínio de classificação binária, tornando necessário adotar outras alternativas de métricas para a avaliação de desempenho da classificação.

A métrica MCC [41] foi originalmente criada para avaliar a performance da predição de estruturas secundárias de proteínas, onde um valor 1 representa ótima performance, e -1 uma ruim. Posteriormente, por focar em resolver o problema de desbalanceamento de classes, passou a ser utilizada na avaliação de problemas de aprendizado de máquina para classificação binária nos trabalhos de [5, 43, 32, 23]. Uma outra alternativa ao uso de acurácia é a métrica F-Score [11], também utilizada para medir desempenho de classificadores binários, onde o valor próximo de 1 representa um ótimo resultado, e próximo de 0 um resultado ruim. Essas duas métricas são utilizadas para avaliar o desempenho dos modelos treinados nesta pesquisa, sendo que focamos no uso de MCC por já ser utilizado em problemas em que há grandes diferenças em distribuição de classes, como no nosso caso. Porém, exibimos também a acurácia e F-Score dos modelos para que possam ser utilizados em trabalhos que queiram utilizar nossos resultados para comparação com essas métricas.

5.1 Configuração Experimental

Os treinamentos, validações e teste de tempo de inferência foram executados usando uma NVIDIA GPU Tesla P100-SXM2-16GB, Intel Xeon CPU E5-2698 v4 2.20GHz, 528 GB de memória. PSPNet foi executada com container de Docker Caffe NVIDIA 17.11. Já a DeepLab foi executada com container de Docker Tensorflow NVIDIA 18.04.

Todos os treinamentos e inferências realizados com PSPNet foram executados em somente uma GPU com as configurações citadas acima. A Tabela 5.3 exhibe que treinar a PSPNet sem *data augmentation* (chamamos de `psp_huskin_no_aug` ou `psp_egohand_no_aug`, de acordo com a base utilizada para treinar) resulta em melhor MCC no conjunto de validação do que com *data augmentation* (`psp_huskin_aug` or `psp_egohand_aug`).

A Deeplab Mobilenet foi treinada com tamanho de *batch* 14 (chamamos de `huskin_14b_Mobilenet` ou `egohand_14b_Mobilenet`) e Xception com *batch* de tamanho 4 (`huskin_4b_Xception` ou `egohand_4b_Xception`) e 14 (`huskin_14b_Xception` ou `egohand_14b_Xception`). Quando o treinamento foi feito com *batch* de tamanho 4, somente uma GPU, com as configurações citadas acima, foi utilizada. Quando utilizado o *batch* de tamanho 14, duas GPUs foram utilizadas ao mesmo tempo. Quando realizada inferência,

Tabela 5.1: Accurácia da PSPNet no conjunto de validação

	Acurácia Média	Desvio Padrão
psp_huskin_aug	96.39	1.463
psp_huskin_no_aug	97.50	0.423
psp_egohand_aug	99.05	0.113
psp_egohand_no_aug	99.23	0.074

somente uma GPU foi usada em ambos os casos.

5.2 Resultados

A Tabela 5.6 demonstra que a versão Huskin Mobilenet é melhor que a Egohand. Entretanto, a melhor configuração Deeplab é egohand_4b_Xception com haskin_4b_Xception com valor muito próximo. Além disso, como podemos observar na Tabela 5.3, qualquer versão de PSPNet possui melhor MCC (melhor acurácia e *F-Score* também, como visto nas Tabelas 5.1 e 5.2), sendo que o treinado com Egohand, sem *data augmentation*, tem o melhor resultado com MCC médio de 0,969.

Observando a Tabela 5.6, também podemos notar que todas as configurações Deeplab possuem MCC baixos, com seus valores muito próximos de 0, indicando que os modelos não são eficientes para classificação binária ou a própria segmentação, sendo quase aleatórios na previsão de classes.

Os resultados qualitativos podem ser vistos na Figura 5.1, sendo que (A) é a imagem original, (B) máscara, (C) deeplab_egohand_14b_mobilenet, (D) deeplab_huskin_14b_mobilenet, (E) deeplab_egohand_14b_xception, (F) deeplab_huskin_14b_xception, (G) psp_egohand_no_aug, (H) psp_huskin_no_aug. Podemos observar que os resultados de PSPNet possuem mais detalhes finos do que Deeplab, apesar de a implementação com Huskin (H) detectar uma mesa como mão, e a implementação com Egohand (G) separar melhor a mão da pessoa da mesa.

A justificativa para essa melhor qualidade na segmentação da PSP pode se dar pela sua quantidade de parâmetros e camadas, que é superior que todas as outras implementações estudadas, o que faz a rede ter mais resultados significativos nas últimas camadas. Seu tamanho é resultado da união das 101 camadas de ResNet com a pirâmide de agrupamento

Tabela 5.2: F-Score da PSPNet no conjunto de validação

	F-Score Média	Desvio Padrão
psp_huskin_aug	0.831	0.0870
psp_huskin_no_aug	0.893	0.0200
psp_egohand_aug	0.961	0.0073
psp_egohand_no_aug	0.969	0.0031

Tabela 5.3: MCC da PSPNet no conjunto de validação

	MCC Média	Desvio Padrão
psp_huskin_aug	0.826	0.0773
psp_huskin_no_aug	0.883	0.0208
psp_egohand_aug	0.957	0.0053
psp_egohand_no_aug	0.965	0.0035

Tabela 5.4: Acurácia da Deeplab no conjunto de validação

	Acurácia Média	Desvio Padrão
huskin_4b_Xception	81.75	0.156
huskin_14b_Mobilenet	81.87	0.164
huskin_14b_Xception	81.59	0.078
egohand_4b_Xception	81.71	0.120
egohand_14b_Mobilenet	81.93	0.117
egohand_14b_Xception	81.26	0.725

no final. Outro fato que pode contribuir para a qualidade é o tamanho de saída do mapa de características antes da aplicação da pirâmide, que consiste em $1/8$ da imagem original de entrada da rede, ou seja, *output stride* de 8, que tem mais detalhes que os de 16 das outras implementações aqui testadas. Além disso, até mesmo a ResNet utilizando a arquitetura DeepLab, exposta em [54], possui menos parâmetros (58,16M parâmetros e 81,0B operações de multiplicação-adição) que a implementação PSP original, por possuir *depthwise separable convolutions* e convoluções dilatadas na pirâmide de agrupamento,

Tabela 5.5: F-Score da Deeplab no conjunto de validação

	F-Score Média	Desvio Padrão
huskin_4b_Xception	0.140	0.0014
huskin_14b_Mobilenet	0.133	0.0019
huskin_14b_Xception	0.142	0.0010
egohand_4b_Xception	0.141	0.0011
egohand_14b_Mobilenet	0.130	0.0018
egohand_14b_Xception	0.138	0.0093

Tabela 5.6: MCC da Deeplab no conjunto de validação

	MCC Média	Desvio Padrão
huskin_4b_Xception	0.043	0.0005
huskin_14b_Mobilenet	0.038	0.0005
huskin_14b_Xception	0.043	0.0002
egohand_4b_Xception	0.044	0.0003
egohand_14b_Mobilenet	0.036	0.0007
egohand_14b_Xception	0.037	0.0150

onde os filtros podem ter diferentes escalas sem aumentar a quantidade de parâmetros.

A utilização de *data augmentation* em todas as configurações DeepLab pode ter atrapalhado seus resultados, pelo fato de o problema em que foram testados consiste somente de imagens de braços saindo da parte inferior da tela e não saindo da parte superior (invertido), que é o que acontece ao realizar a inversão da imagem verticalmente (que faz parte do *data augmentation*). Pode-se verificar que os resultados da PSPNet foram melhores quando não utilizado *data augmentation*.

A implementação de Xception é menor que a PSP (54,17B operações de multiplicação-adição) por também representar a sua pirâmide de agrupamento com convoluções dilatadas. Além disso, a própria quantidade de camadas e a utilização de *depthwise separable convolutions* diminui mais ainda essa quantidade de parâmetros, tornando a rede mais leve, mas gerando perda de qualidade no seu resultado de segmentação.

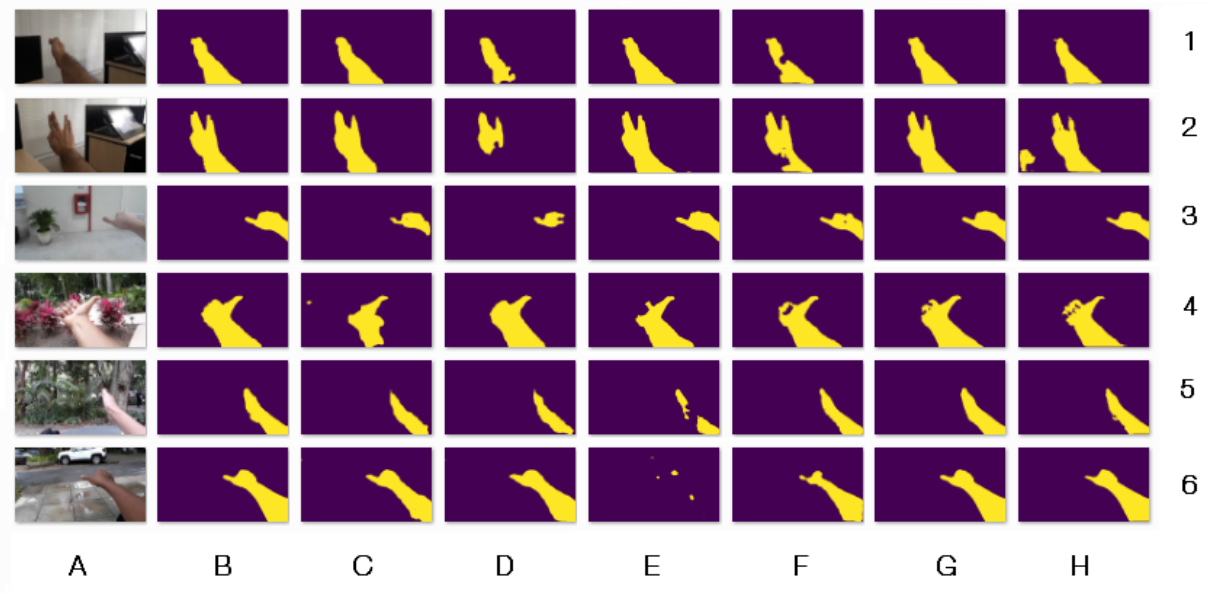


Figura 5.1: Resultados Qualitativos. (A) is the original image, (B) ground truth, (C) deeplab_egohand_14b_mobilenet, (D) deeplab_huskin_14b_mobilenet, (E) deeplab_egohand_14b_xception, (F) deeplab_huskin_14b_xception, (G) psp_egohand_no_aug, (H) psp_huskin_no_aug.

Já a MobileNet aqui apresentada possui consideravelmente menos parâmetros que a Xception (2,11M parâmetros e 2,75B operações), sendo a menor e mais leve de todas, porém também resultando em pior resultado de MCC. Seu tamanho pode ser explicado por não utilizar um módulo ASPP (pois como falado antes, aumenta sua quantidade de parâmetros, aumentando tempo de inferência). Além disso, ela também não possui um módulo decoder, que é responsável por melhorar o mapa de características extraído do encoder na implementação Xception, e seus módulos residuais são *inverted residual block*, diminuindo mais ainda a quantidade de parâmetros pela forma que são estruturadas suas operações (explicadas na sessão 4.2).

Outro fato interessante é o tamanho do *batch* utilizado nos treinamentos das implementações Xception. Com tamanho 4 conseguimos melhor MCC, ou seja, atualizando os pesos de forma mais rápida, sem olhar para muitos dados de treinamentos, alcançamos um melhor modelo, o que geralmente não deveria ocorrer.

Para testar o tempo de inferência, os 322 quadros com resolução 640x360 do conjunto de validação foram utilizados como entrada para as melhores configurações de redes. O tempo de segmentação, que foi medido em segundos da execução desses 322 quadros, podem ser vistos na Tabela 5.7, onde pode-se observar que a versão Deeplab Mobilenet é consideravelmente melhor que as outras, alcançando 38 quadros por segundo. Já PSPNet demonstra que não pode ser utilizada em aplicações de tempo real segmentando cada

Tabela 5.7: Teste de tempo de inferência

	Tempo Médio	Tempo Total
PSP	21,997	7083,085
Mobilenet	0,026	8,577
Xception	0,076	24,475

Tabela 5.8: Teste na base pública GTEA

	MCC	Acurácia	F-Score
PSP	0,520	87,09	0,570
Xception	0,036	82,19	0,134

quadro.

Observando a quantidade de parâmetros e operações das arquiteturas faladas, pode-se explicar que a MobileNet tem melhor tempo de inferência por ser bastante menor que as outras arquiteturas tanto em parâmetros quanto em operações. A PSP perde em tempo de inferência para todas as outras por ter mais operações que todas.

Também foram realizados testes na base pública GTEA [21]. Utilizamos o primeiro conjunto GTEA citado na sessão 2.2.3, no qual contém 7 tipos diferentes de atividades diárias, realizadas por 4 indivíduos diferentes, todas capturadas por câmera acoplada em capacetes utilizados pelos usuários, ou seja, em visão egocêntrica. A Tabela 5.8 exibe as medidas MCC, acurácia e F-Score para as melhores configurações de redes treinadas (Deeplab egohand_4b_Xception e PSP egohand_no_aug), em relação à medida MCC.

Na Figura 5.2 estão exibidos os resultados qualitativos dos testes da Tabela 5.8, onde podemos observar novamente a qualidade superior da rede PSP (treinada sem *data augmentation*) em segmentar, comparada com DeepLab Xception treinada com *batch* de tamanho 4 e *output stride* 16. Testamos também variar o *output stride* de Xception e MobileNet para 8, mas confirmamos o que seus autores encontraram, o tamanho 16 resulta em melhor qualidade de segmentação, segundo todas as métricas aqui mencionadas.

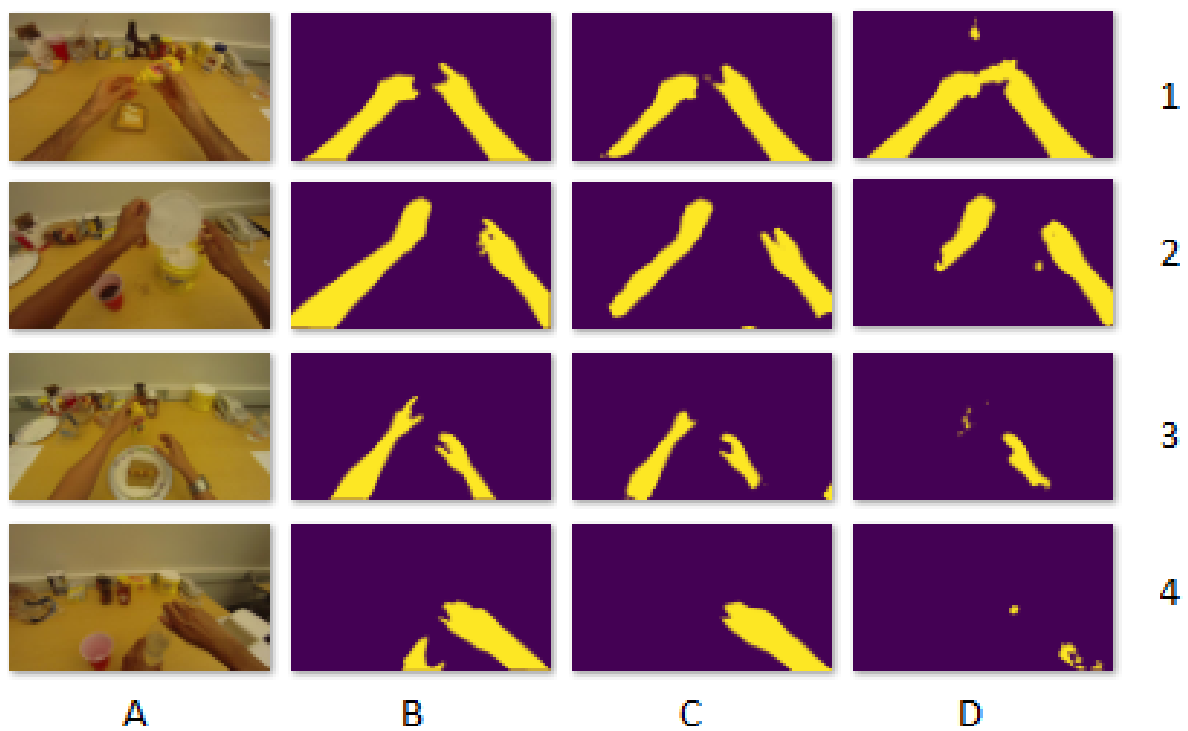


Figura 5.2: Testes na base GTEA. (A) imagem original de 4 sujeitos em atividades diferentes, (B) máscara de braços, (C) PSP, (D) DeepLab.

Capítulo 6

Conclusão

Neste trabalho, propomos uma estratégia de segmentação de braços humanos presentes em imagens em perspectiva de primeira pessoa de aplicações de realidade virtual e mista. Nosso método, quando aplicado para execução rápida, a qualidade de segmentação cai, comparada com quando focado na qualidade, onde ocorre o inverso. Também disponibilizamos o código-fonte e materiais de treinamento para acesso público ¹.

Primeiramente criamos 4041 máscaras para a mesma quantidade de imagens da base FPVRGame [46], gerando assim uma base de dados em visão egocêntrica para treinamento, focada em segmentação de braços humanos, que chamamos Egohand. Criamos também outra base para validação, também em perspectiva de primeira pessoa. A primeira possui imagens capturadas em ambientes internos. A segunda possui imagens em ambientes internos e externos com variados tons de pele e iluminação. Apesar de a motivação da criação dessas bases ter sido para aplicação de realidade mista com visualização dos braços do usuário, ela também pode ser utilizada para treinar redes com o objetivo de segmentar braços para outro objetivo.

Foi também apresentado os *fine-tunings* e configurações das arquiteturas de redes neurais profundas PSPNet e DeepLab, utilizando nossas bases de treinamento e validação. A configuração DeepLabV3+ Xception treinada com 4 *batches* utilizando a base Egohand mostrou-se melhor em tempo de inferência, apesar de não representar um classificador de pixels eficiente segundo os resultados de MCC. Apesar disso, PSPNet mostrou-se ser melhor segundo seu MCC de 0,969, mas inferior em relação ao tempo de inferência, não atingindo tempo real, pois leva mais de 20 segundos para segmentar um quadro. Portanto, a DeepLabV3+ Xception é a configuração que pode ser utilizada na aplicação de realidade virtual ou mista.

¹<https://github.com/indexhever/ahsep>

As configurações DeepLabV3 Mobilenet apresentaram os melhores resultados no quesito tempo de inferência, tornando possível a segmentação de 38 quadros por segundo.

Os melhores resultados de treinamento foram os que utilizaram a base Egohand, sendo melhor que treinamentos realizados em bases não focadas em visão de primeira pessoa, como a Huskin.

Foi verificado também que os resultados da PSPNet foram bons em todas as métricas quando não aplicado *data augmentation*. O que pode ter prejudicado os resultados da DeepLab, que possui *data augmentation* em todas as configurações, e uma das modificações feitas pelo *data augmentation* consiste em inverter verticalmente a imagem. Como os modelos foram todos testados em imagens com os braços do usuário sempre iniciando na parte inferior da imagem, os modelos da DeepLab foram prejudicados por receber no treinamento imagens de braços iniciando na parte superior. Isso pode significar uma possível melhora nos resultados caso os modelos da DeepLab sejam treinados sem inversão vertical no *data augmentation*.

O treinamento em mais de duas GPUs com a PSPNet não foi possível devido a limitações das otimizações dos *frameworks* em que a rede foi implementada nestes cenários. Limitamos a quantidade de GPUs ao treinar a DeepLab para tentar igualar as limitações de treinamento da PSPNet, sendo que DeepLab utiliza *batches* de mais de uma imagem, oferecendo melhor desempenho de treinamento, enquanto a PSPNet possui somente uma imagem para cada *batch* por utilizar bastante memória em uma GPU.

Nosso trabalho, por ter focado em treinar com imagens somente em perspectiva egocêntrica, que representam as situações que o usuário pode presenciar ao utilizar a aplicação de realidade mista, ou seja, que captam as mudanças de iluminação e foco inerentes da movimentação da cabeça do usuário, por esses motivos, o trabalho demonstra estar a frente dos trabalhos relacionados, por possuírem também pontos fracos (como visto na Tabela 3.1) que o nosso não tem.

Para trabalhos futuros, as configurações Deeplab podem ser treinadas por mais iterações para verificar se o MCC aumenta. Além disso, aumentar as bases de treinamento também pode ajudar. Uma próxima etapa importante é a criação de uma aplicação executando a rede neural em imagens capturadas de uma câmera para aplicar a segmentação e testar em tempo real, para após isso, a criação de aplicação com um HMD para testar todo o sistema funcionando. FPVRGame [46] é uma continuação natural da nossa pesquisa: após a renderização dos braços do usuário, a detecção de diferentes movimentos pode ser utilizada para interação no ambiente virtual. Além disso, a captura de ilumina-

ção do ambiente no mundo real para ajustar a iluminação da imagem segmentada, como é feito em [39], pode melhorar a sensação de imersão por deixar a iluminação da imagem do braço do usuário semelhante à iluminação do que está sendo visualizando na aplicação de realidade mista. A comparação com a visualização de representação virtual do corpo do usuário com a imagem real já foi feita em [47], demonstrando que realmente melhora a sensação de imersão e que os estudos na construção desta aplicação devem continuar.

Referências

- [1] ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I. J.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JÓZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D. G.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P. A.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F. B.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR abs/1603.04467* (2016).
- [2] BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv:1511.00561 [cs]* (Nov. 2015). arXiv: 1511.00561.
- [3] BELL, S.; UPCHURCH, P.; SNAVELY, N.; BALA, K. Material Recognition in the Wild with the Materials in Context Database. *arXiv:1412.0623 [cs]* (Dec. 2014). arXiv: 1412.0623.
- [4] BETANCOURT, A.; MORERIO, P.; REGAZZONI, C. S.; RAUTERBERG, M. The Evolution of First Person Vision Methods: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 5 (May 2015), 744–760.
- [5] BOUGHORBEL, S.; JARRAY, F.; EL-ANBARI, M. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLOS ONE* 12, 6 (June 2017), e0177678.
- [6] CASATI, J. P. B.; MORAES, D. R.; RODRIGUES, E. L. L. SFA: A Human Skin Image Database based on FERET and AR Facial Images. In *Anais do VIII Workshop de Visão Computacional* (Rio de Janeiro, 2013).
- [7] CHEN, L.; ZHU, Y.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR abs/1802.02611* (2018).
- [8] CHEN, L.-C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv:1412.7062 [cs]* (Dec. 2014). arXiv: 1412.7062.
- [9] CHEN, L.-C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv:1606.00915 [cs]* (June 2016). arXiv: 1606.00915.

- [10] CHEN, L.-C.; PAPANDREOU, G.; SCHROFF, F.; ADAM, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587 [cs]* (June 2017). arXiv: 1706.05587.
- [11] CHINCHOR, N. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding* (Stroudsburg, PA, USA, 1992), MUC4 '92, Association for Computational Linguistics, pp. 22–29.
- [12] CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. *CoRR abs/1610.02357* (2016).
- [13] DAI, J.; QI, H.; XIONG, Y.; LI, Y.; ZHANG, G.; HU, H.; WEI, Y. Deformable convolutional networks. *CoRR abs/1703.06211* (2017).
- [14] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).
- [15] DOLLAR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 4 (April 2012), 743–761.
- [16] DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv:1603.07285 [cs, stat]* (Mar. 2016). arXiv: 1603.07285.
- [17] EIGEN, D.; FERGUS, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR abs/1411.4734* (2014).
- [18] EVERINGHAM, M.; ESLAMI, S. M. A.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision* 111, 1 (Jan. 2015), 98–136.
- [19] EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [20] EVERINGHAM, M.; VAN GOOL, L.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* 88, 2 (June 2010), 303–338.
- [21] FATHI, A.; REN, X.; REHG, J. M. Learning to recognize objects in egocentric activities. In *CVPR 2011* (June 2011), pp. 3281–3288.
- [22] GARCIA-GARCIA, A.; ORTS-ESCOLANO, S.; OPREA, S.; VILLENA-MARTINEZ, V.; GARCIA-RODRIGUEZ, J. A Review on Deep Learning Techniques Applied to Semantic Segmentation. *arXiv:1704.06857 [cs]* (Apr. 2017). arXiv: 1704.06857.
- [23] GU, Q.; ZHU, L.; CAI, Z. Evaluation measures of the classification performance of imbalanced data sets. In *Computational Intelligence and Intelligent Systems* (Berlin, Heidelberg, 2009), Z. Cai, Z. Li, Z. Kang, and Y. Liu, Eds., Springer Berlin Heidelberg, pp. 461–471.

- [24] GUPTA, S.; ARBELÁEZ, P.; GIRSHICK, R.; MALIK, J. Indoor Scene Understanding with RGB-D Images: Bottom-up Segmentation, Object Detection and Semantic Segmentation. *International Journal of Computer Vision* 112, 2 (Apr. 2015), 133–149.
- [25] HE, K.; SUN, J. Convolutional neural networks at constrained time cost. *CoRR abs/1412.1710* (2014).
- [26] HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385.
- [27] HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]* (Apr. 2017). arXiv: 1704.04861.
- [28] HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat’s striate cortex. *The Journal of Physiology* 148, 3 (Oct. 1959), 574.
- [29] HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology* 195, 1 (1968), 215–243.
- [30] JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).
- [31] KARPATY, A. Cs231 convolutional Neural Networks.
- [32] KOYEJO, O. O.; NATARAJAN, N.; RAVIKUMAR, P. K.; DHILLON, I. S. Consistent binary classification with generalized performance metrics. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2744–2752.
- [33] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90.
- [34] KRÄHENBÜHL, P.; KOLTUN, V. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *arXiv:1210.5644 [cs]* (Oct. 2012). arXiv: 1210.5644.
- [35] LI, C.; KITANI, K. M. Pixel-Level Hand Detection in Ego-centric Videos. pp. 3570–3577.
- [36] LIN, M.; CHEN, Q.; YAN, S. Network in network. *CoRR abs/1312.4400* (2013).
- [37] LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. *CoRR abs/1411.4038* (2014).
- [38] LONG, J.; SHELHAMER, E.; DARRELL, T. Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]* (Nov. 2014). arXiv: 1411.4038.
- [39] MARQUES, B. A. D.; CLUA, E. W. G.; VASCONCELOS, C. N. Deep spherical harmonics light probe estimator for mixed reality games. *Computers & Graphics* 76 (Nov. 2018), 96–106.

- [40] MARTINEZ, A.; BENAVENTE, R. The ar face database. Technical Report 24, CVC, 1998.
- [41] MATTHEWS, B. W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405, 2 (Oct. 1975), 442–451.
- [42] MAURYA, J.; HEBBALAGUPPE, R.; GUPTA, P. Real Time Hand Segmentation on Frugal Headmounted Device for Gestural Interface. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (Oct. 2018), pp. 4023–4027.
- [43] MENON, A.; NARASIMHAN, H.; AGARWAL, S.; CHAWLA, S. On the Statistical Consistency of Algorithms for Binary Classification under Class Imbalance. In *International Conference on Machine Learning* (Feb. 2013), pp. 603–611.
- [44] NOH, H.; HONG, S.; HAN, B. Learning Deconvolution Network for Semantic Segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (Washington, DC, USA, 2015), ICCV '15, IEEE Computer Society, pp. 1520–1528.
- [45] OBDRŽÁLEK, S.; KURILLO, G.; HAN, J.; ABRESCH, T.; BAJCSY, R. Real-time human pose detection and tracking for tele-rehabilitation in virtual reality. *Studies in health technology and informatics* 173 (2012), 320–324.
- [46] OLIVEIRA, E. D. FPVRGame: Deep Learning for Hand Pose Recognition in Real-Time Using Low-End HMD. Master's thesis, UFF, Niterói, Rio de Janeiro, 2019.
- [47] OLIVEIRA, W.; GAISBAUER, W.; TIZUKA, M.; CLUA, E.; HLAVACS, H. Virtual and Real Body Experience Comparison Using Mixed Reality Cycling Environment. In *Entertainment Computing – ICEC 2018* (2018), E. Clua, L. Roque, A. Lugmayr, and P. Tuomi, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 52–63.
- [48] PHILLIPS, P. J.; MOON, H.; RIZVI, S. A.; RAUSS, P. J. The FERET evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 10 (Oct. 2000), 1090–1104.
- [49] PHUNG, S. L.; BOUZERDOUM, A.; CHAI, D. Skin segmentation using color pixel classification: analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 1 (Jan 2005), 148–154.
- [50] QI, X.; LIAO, R.; JIA, J.; FIDLER, S.; URTASUN, R. 3d Graph Neural Networks for RGBD Semantic Segmentation. pp. 5199–5208.
- [51] RAJ, A.; MATURANA, D.; SCHERER, S. Multi-scale convolutional architecture for semantic segmentation. Tech. Rep. CMU-RI-TR-15-21, Carnegie Mellon University, Pittsburgh, PA, October 2015.
- [52] REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs]* (June 2015). arXiv: 1506.02640.

- [53] ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 309–314.
- [54] SANDLER, M.; HOWARD, A. G.; ZHU, M.; ZHMOGINOV, A.; CHEN, L. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR abs/1801.04381* (2018).
- [55] SIDDHARTH; RANGESH, A.; OHN-BAR, E.; TRIVEDI, M. M. Driver Hand Localization and Grasp Analysis: A Vision-based Real-time Approach.
- [56] SIFRE, L.; MALLAT, S. Rigid-motion scattering for image classification. *PhD thesis, Ph. D. thesis 1* (2014), 3.
- [57] SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).
- [58] SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition.
- [59] SLATER, M.; WILBUR, S. A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments. *Presence: Teleoperators and Virtual Environments* 6, 6 (1997), 603–616.
- [60] SOLOMON, C.; BRECKON, T. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, 1st ed. Wiley Publishing, 2011.
- [61] SONG, X.; HERRANZ, L.; JIANG, S. Depth cnns for RGB-D scene recognition: learning from scratch better than transferring from rgb-cnns. *CoRR abs/1801.06797* (2018).
- [62] SONKA, M.; HLAVAC, V.; BOYLE, R. *Image Processing, Analysis, and Machine Vision*, 4 ed. Cengage Learning, Jan. 2014. Google-Books-ID: DAxvCgAAQBAJ.
- [63] SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Highway networks. *CoRR abs/1505.00387* (2015).
- [64] SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. *CoRR abs/1409.4842* (2014).
- [65] SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. *CoRR abs/1512.00567* (2015).
- [66] TU, Z.; BAI, X. Auto-Context and Its Application to High-Level Vision Tasks and 3d Brain Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 10 (Oct. 2010), 1744–1757.
- [67] VASCONCELOS, C. N.; CLUA, E. W. G. Deep Learning - Teoria e Prática. In *Jornadas de Atualização em Informática 2017* (Porto Alegre/RS, July 2017), Sociedade Brasileira de Computação - SBC.

- [68] VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (2001), vol. 1, pp. I–511–I–518 vol.1.
- [69] VIOLA, P.; JONES, M. J. Robust Real-Time Face Detection. *International Journal of Computer Vision* 57, 2 (May 2004), 137–154.
- [70] VODOPIVEC, T.; LEPETIT, V.; PEER, P. Fine Hand Segmentation using Convolutional Neural Networks. *arXiv:1608.07454 [cs]* (Aug. 2016). arXiv: 1608.07454.
- [71] WANG, C.-F. A Basic Introduction to Separable Convolutions, Aug. 2018.
- [72] YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? *CoRR abs/1411.1792* (2014).
- [73] YU, F.; KOLTUN, V. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv:1511.07122 [cs]* (Nov. 2015). arXiv: 1511.07122.
- [74] ZEILER, M. D.; FERGUS, R. Visualizing and Understanding Convolutional Networks. In *Computer Vision – ECCV 2014* (Sept. 2014), Lecture Notes in Computer Science, Springer, Cham, pp. 818–833.
- [75] ZEILER, M. D.; TAYLOR, G. W.; FERGUS, R. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision* (Barcelona, Spain, Nov. 2011), IEEE, pp. 2018–2025.
- [76] ZHAO, H.; SHI, J.; QI, X.; WANG, X.; JIA, J. Pyramid Scene Parsing Network. *arXiv:1612.01105 [cs]* (Dec. 2016). arXiv: 1612.01105.
- [77] ZITNICK, C. L.; DOLLÁR, P. Edge Boxes: Locating Object Proposals from Edges. In *Computer Vision – ECCV 2014* (2014), D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 391–405.