CENTRO UNIVERSITÁRIO FEI

HENRIQUE XAVIER GOULART

CLASSIFICAÇÃO DE CENAS EM IMAGENS ATRAVÉS DA ARQUITETURA COGNITIVA LIDA

São Bernardo do Campo

2020

HENRIQUE XAVIER GOULART

CLASSIFICAÇÃO DE CENAS EM IMAGENS ATRAVÉS DA ARQUITETURA COGNITIVA LIDA

Dissertação de mestrado, apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Dr. Paulo Sérgio Silva Rodrigues e co-orientado pelo Prof. Dr. Guilherme Alberto Wachs Lopes.

São Bernardo do Campo

2020

Xavier Goulart, Henrique.
Classificação de Cenas em Imagens Através da Arquitetura Cognitiva LIDA / Henrique Xavier Goulart. São Bernardo do Campo, 2020.
93 p. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Paulo Sergio Silva Rodrigues.
Coorientador: Prof. Dr. Guilherme Alberto Wachs Lopes.

1. LIDA. 2. SVM. 3. redes neurais. 4. classificação de cenas. I. Sergio Silva Rodrigues, Paulo, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Henrique Xavier Goulart **Matrícula:** 118110-6

Título do Trabalho: Classificação de cenas em imagens através da arquitetura cognitiva lida..

Área de Concentração: Processamento de Sinais e Imagens

Orientador: Prof. Dr. Paulo Sergio Silva Rodrigues

Data da realização da defesa: 16/03/2020 **ORIGINAL ASSINADA**

Avaliação da Banca Examinadora:

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Paulo Sergio Silva Rodrigues	Ass.: _____
Prof. Dr. Roberto Baginski Batista Santos	Ass.: _____
Prof. Dr. Gilson Antônio Giraldi	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

À minha família.

AGRADECIMENTOS

O mestrado é a fase que um pesquisador busca alcançar o atual limite do conhecimento humano em um tema, já vislumbrando ultrapassá-lo. Este é um momento de colocar mais um tijolo na parede do conhecimento científico humano; e para chegar até este momento, muitas pessoas contribuíram das mais diversas maneiras para a elaboração deste trabalho.

Primeiramente quero agradecer à minha companheira Eladia Penteado, por me apoiar durante esta jornada. Seu apoio se deu das mais abrangentes formas, acreditando no meu potencial, em muitas ocasiões, até mais do que eu mesmo; incessantemente, me relembrando da importância do título de mestre na minha vida. Eladia, espero passar toda a minha vida retribuindo todo o esforço que você despendeu em mim.

Quero agradecer também à minha mãe, Eliana Xavier Goulart, por ter sempre cuidado e se dedicado da minha educação desde pequeno, e me guiado até este momento. Também quero agradecer ao meu pai, Eliseu Goulart, por me apoiar incansavelmente, independentemente das dificuldades, para que eu pudesse estudar e chegar até aqui. Amo muito vocês dois! E me espelho em vocês para continuar melhorando como pessoa.

Sou grato também ao Prof. Dr. Guilherme Alberto Wachs Lopes, que me orientou desde a iniciação científica, TCC e, agora, mestrado. Contribuindo muito para formar o pesquisador que sou hoje, sempre ensinando a pesquisar, escrever e conseguir transmitir ideias de forma clara. Desta forma, seguiu sempre me motivando a ir cada vez mais longe.

Outra pessoa que teve um papel fundamental neste trabalho, foi o Prof. Dr. Paulo Sérgio Silva Rodrigues, que orientou este trabalho e cuidadosamente o refinou para melhorá-lo muito. Sempre instigando o debate de ideias para contribuir com o amadurecimento das asserções propostas e para a produção de um trabalho apropriado ao título de mestre.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

RESUMO

Na área de visão computacional, dentre as diversas linhas existentes, há aquela denominada de generalista. Esta vertente consiste em estudos amplos, incluindo muitos elementos e variáveis; nesse caso, inúmeras aplicações podem ser desenvolvidas dentro de um mesmo estudo. Um dos principais problemas estudados na área de visão computacional, seguindo a linha generalista, é conhecido como classificação de cenas. Na literatura, esse problema é geralmente enfrentado com o uso de técnicas de *Deep Learning* em combinação com outras técnicas de classificação. Em contrapartida, uma linha de pesquisa conhecida como arquiteturas cognitivas, que embora seja pouco explorada em visão computacional, vem sendo estudada nas últimas décadas buscando mesclar conceitos da neurociência e da ciência da computação. Neste trabalho, será apresentado um modelo que agrega a arquitetura cognitiva Learning Intelligent Distribution Agent (LIDA) com métodos já utilizados na área, visando estudar os aspectos poucos explorados até o momento, que são proporcionadas por esta união. Dentre os métodos existentes na área, serão utilizados o *Support Vector Machine* (SVM), *Multilayer Perceptron* (MLP) e Redes Neurais Convolucionais (CNN). Este trabalho contribuiu para a união pouco explorada descrita aqui, e também serve como base pra futuros trabalhos que estudem problemas através desta união. Outra contribuição é que o presente trabalho apresentou resultados similares ao estado-da-arte em bases de dados frequentemente utilizadas na área, como é o caso do MIT Indoor 67 (85,9% de acurácia) e SUN 397 (69,4% de acurácia), que são bases de cenas naturais coloridas segregadas entre diversas classes.

Palavras-chave: LIDA, SVM, redes neurais, classificação de cenas.

ABSTRACT

In the computational vision field, among other lines of research, there is one line known by generalist. That line of research includes deeper studies with many subjects and variables; it means more complex problems, therefore more practical applications can be developed by using these studies. One of the main computational vision tasks, following the generalist research, is the scene classification. In literature, this problem is solved by methods that use Deep Learning together with other classification techniques. However, there is a research field known by cognitive architectures, although are few studies, has been investigated in the last decades combining concepts of neuroscience and computational science. The present work will present a model that combines the cognitive architecture Learning Intelligent Distribution Agent (LIDA) with methods proposed to solve the problem, aiming to study the possibilities relatively unexploited that are provided by that combination. This work use Support Vector Machine (SVM), Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN) to handle the scene classification task. The present work contributes to effort the combination of techniques stated before, besides it serves as a base for future studies in both fields. Furthermore, this work shows similar results of accuracy to the state-of-the-art in datasets often used: MIT Indoor 67 (85.9%) and SUN 397 (69.4%).

Keywords: LIDA, SVM, neural networks, scene classification.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A área de visão computacional busca modelar o mundo visual e sua rica complexidade através de técnicas matemáticas e modelos computacionais (SZELISKI, 2010). A pesquisa nessa área segue duas principais linhas epistemológicas: as linhas específicas e as generalistas (WACHS-LOPES, 2016). Na primeira linha, as pesquisas propõem o estudo de um tema específico; isto é, se aprofundar ao máximo possível visando uma aplicação, como a detecção de dígitos em códigos de barras ou QR code. Por sua vez, na linha generalista, como a interpretação de cenas, o estudo é mais amplo, incluindo mais elementos e variáveis; nesse caso, inúmeras aplicações podem ser desenvolvidas dentro de uma mesma linha. Ao longo dos anos, uma gama de aplicações surgiu, abrangendo áreas além da Ciência da Computação, tais como engenharia, física, medicina, entre outras (WACHS-LOPES, 2016). Até os dias atuais, tais aplicações são desenvolvidas seguindo diversas linhas de pesquisa, como classificação de câncer de pele (ESTEVA et al., 2017), navegação de veículos aéreos não tripulados (KANELLAKIS; NIKOLAKOPOULOS, 2017), vigilância (SZELISKI, 2010), entre muitas outras.

Ainda dentro da linha de pesquisa generalista, a área de classificação de cenas em imagens é frequentemente explorada dentro da visão computacional (LIANG; HU, 2015; WOHLHART; LEPETIT, 2015). Essa linha é fundamental para diversas aplicações da robótica no mundo real (EITEL et al., 2015), como também para muitos outros projetos desenvolvidos em visão computacional, como o sistema de navegação visual em carros autônomos. Além disso, as execuções de classificações dessa natureza não são triviais com um modelo computacional, uma vez que são realizados em ambientes não controlados, nos quais não é possível controlar a iluminação, posicionamento de câmera, ajuste de foco, entre outros fatores que podem interferir de maneira direta na aplicação. É por esse motivo que essa área ainda é foco de muitas pesquisas.

Majoritariamente, nos últimos anos, as abordagens elaboradas para a classificação de cenas utilizam metodologias de inteligência artificial biologicamente inspiradas. Dentre as mais conhecidas estão as redes neurais (HINTON; SALAKHUTDINOV, 2006; LECUN; BENGIO; HINTON, 2015); algoritmos genéticos (ANKENBRANDT; BUCKLES; PETRY, 1990); e otimização por enxame de partículas (ZHAO; QI, 2011). Além dessas abordagens, métodos estatísticos baseados em aprendizagem de máquina, como o *Support Vector Machine* (SVM), também foram empregados para tratar o problema de classificação de cenas (SICRE et al., 2017).

Seguindo as linhas de pesquisa dos modelos biologicamente inspirados, as arquiteturas cognitivas são metodologias que há muitas décadas estão sendo investigadas. Tais metodologias raramente foram utilizadas em aplicações de classificação de cenas. Tais arquiteturas (também conhecidas como modelos de Inteligência Artificial Generalistas) mostraram ser capazes de lidar com diversos problemas, como mostra Kotseruba e Tsotsos (2018). Essa abordagem se propõe a sintetizar a cognição humana computacionalmente, baseando-se na ciência cognitiva e na neurociência, como define o autor. Desta forma, tais modelos fogem dos usuais modelos biologicamente inspirados que se atêm somente na sintetização que toma um ponto de vista "anatômico", como os neurônios empregados em uma rede neural artificial (*Deep Learning*).

De acordo com Russell e Norvig (2009), a área de Inteligência Artificial pode ser explorada de quatro maneiras distintas: modelos que pensam como humanos, que agem como humanos, que pensam racionalmente e os que agem racionalmente. As arquiteturas cognitivas podem seguir, as categorias que pensam como humanos e que agem como humanos (KOTSERUBA; TSOTSOS, 2018), simultaneamente; por isso são mais generalistas. Um ponto que Kotseruba e Tsotsos (2018) levanta é o fato que um modelo cognitivo artificial deve acertar e errar como um ser humano em uma situação similar. Devido a isso, as arquiteturas cognitivas oferecem uma abordagem distinta dos modelos biologicamente inspirados; resolvendo problemas de forma diferente e mais similar a aspectos relacionados a um ser humano.

Dentre as arquiteturas mais conhecidas, a Learning Intelligent Distribution Agent (LIDA), proposta por Franklin et al. (2014), é uma arquitetura cognitiva que segue o princípio de agentes autônomos (humanos, robôs, etc): percebe o ambiente continuamente, interpreta as informações percebidas, e finalmente realiza uma ação. Esta arquitetura proporciona componentes para atenção, seleção de ação e um aprendizado similar ao humano. Experimentalmente, o autor obteve resultados mostrando que a LIDA se comporta de forma semelhante a pessoas nas tarefas testadas.

O restante do trabalho está dividido da seguinte maneira. O Capítulo 2 destaca a revisão bibliográfica de classificações em imagens e das arquiteturas cognitivas. O Capítulo 3 apresenta os conceitos fundamentais envolvidos no modelo proposto neste trabalho. O Capítulo 4 descreve o método utilizado para desenvolver o modelo. O Capítulo 5 mostra os resultados obtidos dos experimentos. O Capítulo 6 destaca tópicos de disscussão referente aos resultados, como também sobre o próprio trabalho. O Capítulo 7 apresenta as conclusões sobre o trabalho, aplicações, tal como os trabalhos futuros.

1.1 OBJETIVO PRINCIPAL

Este trabalho tem como objetivo principal desenvolver um modelo para a classificação de cenas através de *bag of features* com base na arquitetura LIDA.

1.2 OBJETIVO SECUNDÁRIO

O presente trabalho busca uma união entre a arquitetura LIDA e alguns métodos utilizados para tratar problemas da visão computacional: as redes neurais e o *Support Vector Machine*.

2 TRABALHOS RELACIONADOS

O objetivo deste trabalho é classificar cenas tendo como base a arquitetura LIDA e, além disso, unir esta arquitetura com modelos já utilizados na área (redes neurais e o *Support Vector Machine*). Sendo assim, serão apresentados neste capítulo, os trabalhos relacionados que visaram os mesmos objetivos publicados nos últimos anos, ou que também possuam objetivos semelhantes, como classificações, segmentações e detecções de objetos em imagens ou vídeos; cujos objetivos são a classificação de regiões de imagens de acordo com o contexto da cena relacionada. Além disso, uma vez que este trabalho envolve arquiteturas cognitivas, será apresentada uma revisão bibliográfica dessas arquiteturas, que ainda representam uma área pouco explorada comparada às outras como *Deep Learning*.

A Seção 2.1 apresenta os principais trabalhos relacionados, contendo uma breve descrição da metodologia utilizada e dos resultados obtidos por esses trabalhos. Além disso, os trabalhos são apresentados na Tabela 1 em ordem cronológica, permitindo a visualização da evolução dos principais tópicos envolvidos no tema deste trabalho. Na Tabela 2 são apresentados os resultados destes trabalhos nas duas bases, de classificação de cenas, mais utilizadas na literatura recente (MIT Indoor 67 e SUN 397). Por outro lado, a Seção 2.2 apresenta dados de arquiteturas cognitivas propostas nas últimas décadas, incluindo a arquitetura que será utilizada neste trabalho (LIDA).

2.1 TRABALHOS

Tabela 1 – Sumário de trabalhos relacionados com o *checklist* de características de pesquisa.

	Tipo de Inferência				*Método Utilizado*		
Trabalho	**Detecção de Objetos**	**Classificação de Objetos**	**Classificação de Cenas**	**Review/Survey**	**Rede Neural**	**Arquitetura Cognitiva**	**SVM**
Ng et al. (2012)	X				X	X	X
Krizhevsky, Sutskever e Hinton (2012)		X			X		
Bulò e Kontschieder (2014)		X					
Kontschieder et al. (2014)	X						
Zhou et al. (2014)			X		X		X
Dixit et al. (2015)			X		X		X
Eitel et al. (2015)	X				X		
Liang e Hu (2015)		X			X		
Simonyan e Zisserman (2015)	X	X			X		X
Herranz, Jiang e Li (2016)			X		X		X
Madl et al. (2016)	X				X	X	
Garcia-Gasulla et al. (2017)					X	X	
Tamaazousti, Borgne e Hudelot (2017)		X			X		X
Wang et al. (2017)			X		X		X
Tian et al. (2018)	X				X		
Zhen e Zhang (2018)			X		X		
Cheng et al. (2018)			X		X		X
Zhao e Larson (2018)			X		X		X
Kotseruba e Tsotsos (2018)				X		X	
Liu, Tian e Xu (2019)			X		X		

Fonte: Autor

Ng et al. (2012) propuseram uma metodologia para o entendimento de cenas, mais especificamente, classificação de objetos. O trabalho empegou uma arquitetura cognitiva denominada DSO, que foi desenhada com base em 5 regiões centrais do cérebro humano: Córtex Frontal, Percepção (utilizando SVM e redes neurais *Multilayer Perceptron*), Sistema Límbico, Córtex Associativo e Córtex Motor. A proposta foi testada em uma base de 138 imagens supervisionadas, coletadas em um campo militar, obtendo mais de 90% de acerto em algumas classes de objetos. A metodologia também foi testada na base MSTAR (U.S. AIR FORCE, 2012), alcançando 91% de acerto em determinadas categorias.

Krizhevsky, Sutskever e Hinton (2012) introduziram um trabalho com o propósito de classificar objetos. Para desempenhar tal tarefa, propuseram a ImageNet-CNN, uma rede neural convolucional. A rede neural possuía, além das camadas de convolução, camadas de *Max Pooling* e camadas *fully connected*. O modelo foi treinado e testado na base de dados que leva o nome, a ImageNet (DENG et al., 2009) contando com suas 1000 classes. Os resultados de classificação no Top-1 (a classificação correta foi apontada como mais provável pelo modelo) alcançou 37,5% de taxa de erro, enquanto no Top-5 (a classificação correta foi apontada entre as 5 classes como mais prováveis pelo modelo) 17%.

Bulò e Kontschieder (2014) apresentaram um modelo para classificar objetos, introduzindo uma nova abordagem denominada *Neural Decision Forest*, que é uma derivação das Árvores de Decisão Neurais. Neste método, cada nó da floresta é uma rede neural rMLP. A imagem de entrada é dividida em partes menores de modo a reduzir a complexidade do problema e permitindo que cada nó processe menos informações e, consequentemente, reduzindo o tempo devido ao paralelismo de nós. O modelo foi testado na base Etrims8 (KORC; FÖRSTNER, 2009), adquirindo 80,8% de acerto considerando as 8 classes disponíveis; na base de vídeos CamVid (BROSTOW et al., 2008; BROSTOW; FAUQUEUR; CIPOLLA, 2008) alcançou 82,1% com as 32 classes; e na base LFW (HUANG et al., 2007) obteve 95,4% com 8 classes.

Kontschieder et al. (2014) divulgaram um trabalho para a segmentação e detecção de objetos, propondo o uso de *Random Forests* para estruturar *labels* a partir de imagens, sendo que, cada *label* (que é uma região da imagem) descreve um padrão local. Deste modo, quando estruturados os *labels* expressam padrões globais de uma imagem. A metodologia foi testada na base de dados CamVid, obtendo 83,8% de acerto na segmentação; na base MSRCv2 (SHOTTON et al., 2006), adquiriu 70% de acerto na segmentação das 21 classes; na base KAIST Hanja2 (NOWOZIN et al., 2011) obteve 79,36% de acurácia na detecção; a base TU Darmstadt

(ANDRILUKA; ROTH; SCHIELE, 2008) foi utilizada como treino, para um experimento, e a base Campus Scene (BARINOVA; LEMPITSKY; KOHLI, 2010) para teste, alcançando 98% de acerto na detecção.

Zhou et al. (2014) introduziram uma base de dados focada na classificação de cenas, a base Places. Além disso, utilizaram essa base como treino de uma rede neural convolucional, denominada Places-CNN. A proposta do trabalho visa pegar a saída de uma das camadas intermediárias da rede neural e reestruturar como um vetor de *features* (*bag of features*). A rede neural foi utilizada para gerar as *bag of features* das imagens, em diferentes escalas, de outras bases de dados utilizando o *transfer learning*. Em seguida, as imagens foram classificadas utilizando um *Support Vector Machine* (SVM) com *kernel* linear. A classificação, combinando as *bag of features* da Places-CNN com as da ImageNet-CNN, alcançaram até 70,8% de acerto na base MIT Indoor 67 (QUATTONI; TORRALBA, 2009) com 67 classes; na base SUN 397 (XIAO et al., 2010) conseguiu 53,83% com 397 classes; na base Scenes-15 (LAZEBNIK; SCHMID; PONCE, 2006) 91,59% com 15 classes; enquanto na base SUN Attribute (PATTERSON; HAYS, 2012) 91,56% com 707 classes; na Caltech-101 (FEI-FEI; FERGUS; PERONA, 2007) 84,97% com 101 classes; na Caltech-256 (GRIFFIN; HOLUB; PERONA, 2007) conseguiu 65,06% com 256 classes; na base Action40 (YAO et al., 2011) 55,28% com 40 classes; e na base de dados Event8 (LI; FEI-FEI, 2007) acertou 94,22% com 8 classes.

Dixit et al. (2015) propuseram um modelo para classificar cenas. O trabalho utilizou uma rede neural convolucional (CNN) pré-treinada na base de dados ImageNet (ImageNet-CNN). Foi feita a extração de *bag of features* na camada *softmax* e os vetores foram sumarizados com vetores de Fisher. Utilizando o *transfer learning* da ImageNet-CNN em conjunto aos vetores produzidos por uma Places-CNN, os resultados de acerto na base MIT Indoor 67 chegaram a 79%; enquanto na base SUN 397 alcançaram 61,72%.

Eitel et al. (2015) apresentaram um modelo para classificar objetos e cenas concatenando duas redes neurais convolucionais ImageNet-CNN (utilizando a arquitetura CaffeNet (JIA et al., 2014)), cada uma com um propósito distinto. A primeira rede neural processa a informação de cores nos 3 canais, enquanto a segunda processa a informação de profundidade de cada pixel (tal informação pode ser proveniente de sensores em um robô, por exemplo). O modelo proposto foi avaliado na base de dados RGB-D Object Dataset (LAI et al., 2011), alcançando 84,7% de acurácia com 51 classes; e também foi testado na base RGB-D Scenes Dataset (LAI et al., 2012), obtendo 82,1% com 6 classes.

Liang e Hu (2015) divulgaram um trabalho para classificação de objetos, introduzindo um novo tipo de rede neural, chamada rede neural convolucional recorrente (RCNN). Essa rede neural permite o uso de informações temporais nos filtros de convolução, ampliando a capacidade da rede em detectar padrões. A metodologia foi avaliada na base CIFAR10 (KRIZHEVSKY; HINTON, 2009), alcançando 7,09% de taxa de erro com 10 classes; na base CIFAR100 (KRIZHEVSKY; HINTON, 2009) 31,75% com 100 classes; na base MNIST (LECUN et al., 1998) atingiu 0,31% de taxa de erro com 10 classes; e por fim, na base de dados SVHN (NETZER et al., 2011) obteve 1,77% com 10 classes.

Simonyan e Zisserman (2015) desenvolveram um trabalho em que compararam diversos tamanhos de redes neurais convolucionais na tarefa de classificação de objetos, utilizando a arquitetura que os mesmos denominaram de VGG. Foram propostos diversas combinações de profundidade da rede neural, concatenação de diferentes redes neurais, reescalar as imagens da base de dados e recortar regiões de interesse. Na melhor combinação, a rede proposta atingiu 23,7% de taxa de erro top-1 na base ImageNet e 6,8% no top-5; na base VOC-2007 (EVERINGHAM et al., 2007) alcançou 89,7% de acerto com 20 classes; na base VOC-2012 (EVERINGHAM et al., 2012) 89,3% de acerto com 20 classes; por sua vez, na base Caltech-101, o resultado foi de 92,7%; e finalmente, na base Caltech-256, atingiu 86,2%.

Herranz, Jiang e Li (2016) propuseram um modelo para classificar cenas, visando corrigir uma deficiência existente no modelo apresentado por Zhou et al. (2014). Para isso, também utilizavam uma rede neural convolucional híbrida (ImageNet-CNN em conjunto com a Places-CNN, utilizando a arquitetura AlexNet), porém utilizava escalas de imagens específicas para cada uma das redes propostas; isto é, as menores escalas foram processadas pela ImageNet-CNN e as maiores pela Places-CNN. Essa diferenciação de informações locais e globais aumentou a acurácia do modelo, uma vez que na base Scenes-15 alcançou 94,51% de acerto contra 91,59% do modelo de Zhou et al. (2014); na base MIT Indoor 67 chegou a 80.97% contra 70,8%; e na base SUN 397 66,26% contra 53,83%.

Madl et al. (2016) introduziram um modelo baseado na arquitetura LIDA visando uma aplicação do mundo real. O modelo consistia em um sistema de navegação de robô que levava em consideração as informações espaciais do ambiente. Além disso, o modelo processava as informações visuais utilizando uma rede neural convolucional que extraía *features* e as interpretavam como nós de um grafo. Esta funcionalidade era uma extensão da memória perceptual associativa (PAM), assim o módulo da LIDA foi renomeado para EPAM. A rede neural do EPAM foi treinada para detectar e classificar objetos para que, posteriormente, a memória es-

pacial pudesse atuar em tais informações colhidas. Os experimentos revelaram que o modelo proposto replicou fielmente, em um ambiente virtual, os padrões humanos, segundo os autores.

Garcia-Gasulla et al. (2017) apresentaram um trabalho onde estudam a visualização gráfica de *features* de imagens. A metodologia proposta parte dos conceitos já estabelecidos no processamento de linguagem natural (NLP) e os adapta para a visão computacional. Os estudos foram feitos utilizando uma rede neural convolucional (arquitetura GoogLeNet) treinada na base de dados ImageNet, cujos experimentos visaram mostrar histogramas das similaridades das *bags of features*, clusterização de amostras da base, e o funcionamento de aritméticas de vetores. Além disso, os autores ainda discutem as implicações do estudo nas arquiteturas cognitivas.

Tamaazousti, Borgne e Hudelot (2017) propuseram um modelo para classificar objetos, para o qual utilizaram duas redes neurais convolucionais; uma das redes neurais (Net-G) tinha o objetivo de detectar padrões globais dos objetos, enquanto a outra focava em padrões locais (Net-S). Novamente, ambas redes neurais exportavam *bags of features* para posteriormente serem classificadas por um SVM. O modelo (implementado na arquitetura VGG) obteve nos testes, 92% de acerto na base de dados Caltech-101, 80,9% na base Caltech-256, 87,5% na VOC-2007 e, por fim, 86,1% na VOC-2012.

Wang et al. (2017) estudaram os efeitos da aplicação de multi-escalas em rede neurais convolucionais para classificar cenas e objetos. De forma semelhante ao trabalho de Krizhevsky, Sutskever e Hinton (2012), utilizaram as redes neurais para processar as imagens e também classificá-las. Uma segunda contribuição da pesquisa foi guiar o conhecimento das classes; isto é, baseado em uma matriz de confusão, gerar classes (objetos ou cenas) artificiais que não possuam ambiguidades entre si, as quais as redes neurais têm mais facilidade para associar aos padrões de imagens. Assim, a proposta foi testada novamente utilizando *transfer learning*, na base MIT Indoor 67 conseguindo 86,7% de acerto e 72% na SUN 397.

Tian et al. (2018) apresentaram um estudo que visava gerar base de dados artificiais para auxiliar a detecção de objetos. As imagens artificiais foram geradas com motores gráficos em conjunto com dados de mapas urbanos e uma API de geração de cidades artificiais. Com isso, foram produzidas imagens com uma variabilidade de informação controlada de ambientes urbanos. Somando essas imagens com as imagens providas por uma base de dados real, uma RCNN (com arquitetura ResNet) tinha uma maior gama de padrões para ser treinada e desempenhar a detecção de objetos. Juntando as imagens artificiais com algumas das bases VOC e COCO (foram utilizadas 3 classes comuns as duas bases), o modelo atingiu 75,4% de acerto na classe "carro"; e com a base KITTI (GEIGER; LENZ; URTASUN, 2012) 79,8% na mesma classe.

Zhen e Zhang (2018) propuseram um modelo para a classificação de cenas focando em informações espaciais. A metodologia consiste em dividir as imagens de entrada por *grids* (2×2, 3×3, ...) de modo a extrair regiões das imagens. Através de redes neurais convolucionais, foram extraídas *bags of features* de cada região para então serem assimiladas umas às outras por camadas *fully connected*, conectando assim todas as *features* e gerando resultado consolidado. Esse modelo foi sobre-treinado (*transfer learning*) na base MIT Indoor 67 alcançando 75,73% de acerto e na base SUN 397 acertou 56,58%.

Cheng et al. (2018) apresentaram um trabalho visando classificar cenas utilizando redes neurais e análises estatísticas. O trabalho conta com a utilização de multi-escalas, ImageNet-CNN e Places-CNN em um *pipeline*. Porém, a inovação deste trabalho foi a adição de uma inferência estatística para determinar características discriminantes de objetos para descrever a cena, Assim, utilizada uma rede de co-ocorrências de objetos juntamente com a regra de Bayes. A metodologia proposta foi testada na base MIT Indoor 67 alcançando 86,76% de acerto; na base SUN 397 obteve 73,41%; e na base Scenes-15 95,88%.

Zhao e Larson (2018) introduziram uma abordagem para a classificação de cenas determinando regiões discriminativas da imagem de forma adaptativa. Semelhantemente ao trabalho de Zhen e Zhang (2018), a partir dessas regiões foram extraídas *bags of features* que posteriormente foram concatenadas de modo a gerar uma reposta unificada da imagem. Essa abordagem foi testada (utilizando a arquitetura VGG) na base de dados SUN 397 atingindo 73,59% utilizando a arquitetura ResNet.

Liu, Tian e Xu (2019) propuseram um modelo para classificar cenas que conta com um tratamento (*data augmentation*). O modelo utiliza uma rede neural convolucional (arquitetura ResNet) para extrair *features* treinada na base combinada de imagens providas da ImageNet como também da base Places. Diferentemente dos modelos anteriores, esse extrai *features* a cada duas camadas residuais da Resnet-18 e concatena todas em uma *bag of features* da imagem para uma posterior classificação com uma camada *softmax*. Além disso, o trabalho conta com uma base dados mais ampla através da criação de novas imagens a partir das existentes (*data augmentation*), levando em consideração uma regra de avaliação para as imagens geradas em relação às originais (desta forma decidindo se as imagens geradas devem ou não serem adicionadas às demais). O modelo foi testado na base MIT Indoor 67 alcançando 94,05% de acerto com *data augmentation* e 74,63% sem essa funcionalidade; na base SUN 397 obteve 85,21% com *data augmentation* e 65,46% sem.

Tabela 2 – Sumário de resultados nas bases mais utilizadas recentemente.

Trabalho	**MIT Indoor 67**	**SUN 397**
Ng et al. (2012)		
Krizhevsky, Sutskever e Hinton (2012)		
Bulò e Kontschieder (2014)		
Kontschieder et al. (2014)		
Zhou et al. (2014)	70,8%	53,83%
Dixit et al. (2015)	79%	61,72%
Eitel et al. (2015)		
Liang e Hu (2015)		
Simonyan e Zisserman (2015)		
Herranz, Jiang e Li (2016)	80,97%	66,26%
Madl et al. (2016)		
Garcia-Gasulla et al. (2017)		
Tamaazousti, Borgne e Hudelot (2017)		
Wang et al. (2017)	86,7%	72%
Tian et al. (2018)		
Zhen e Zhang (2018)	75,73%	56,58%
Cheng et al. (2018)	86,76%	73,41%
Zhao e Larson (2018)		73,59%
Liu, Tian e Xu (2019)	94,05% / 74,63%	85,21% / 65,46%

Fonte: Autor

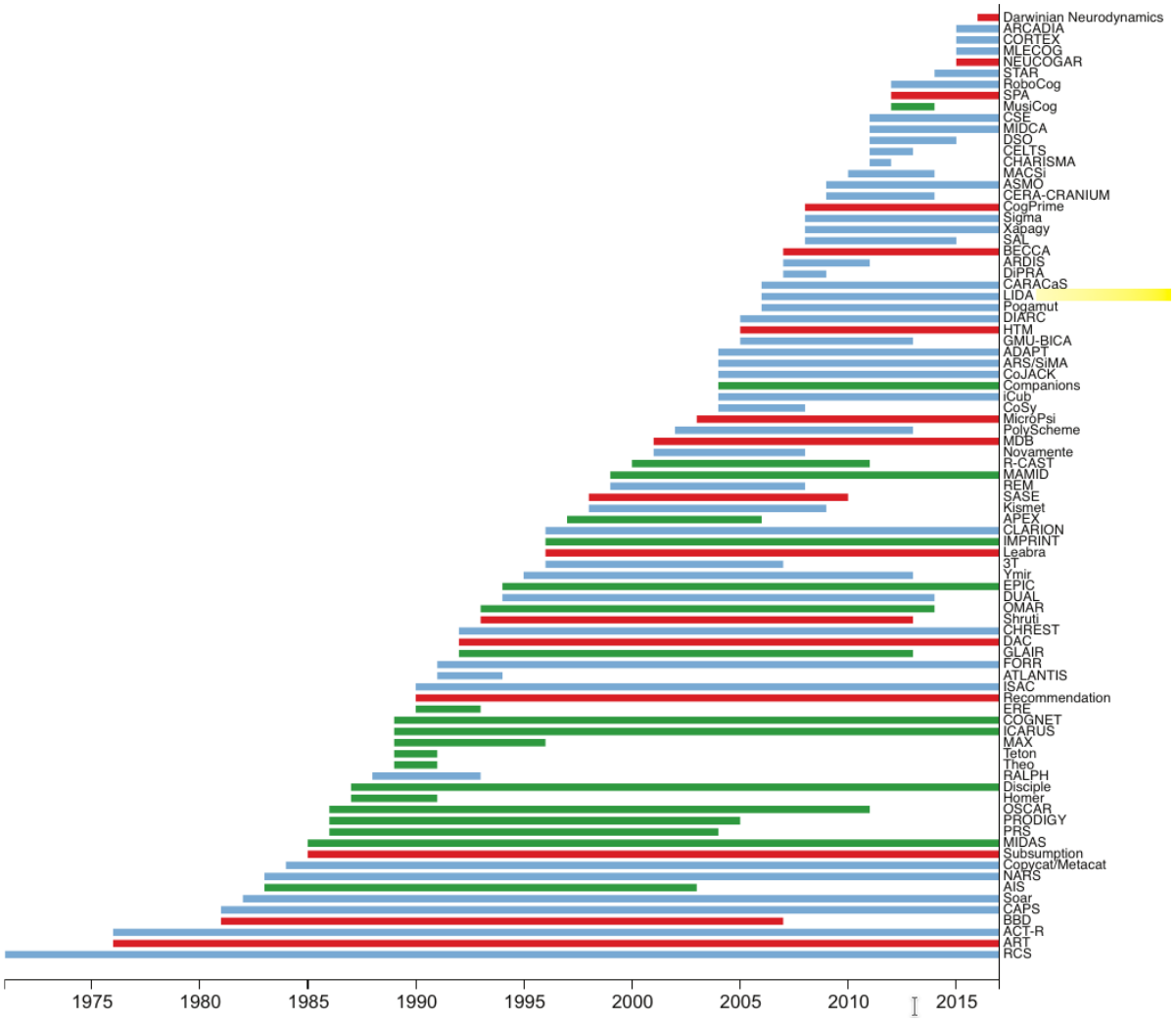
2.2 ARQUITETURAS COGNITIVAS

Esta seção apresenta as arquiteturas cognitivas relacionadas a este trabalho e suas aplicações práticas. Embora tenham sido propostas quase que exclusivamente por pesquisadores da área de ciências exatas (como o ACT-R, proposta por Anderson, Matessa e Lebiere (1997)), tais arquiteturas são tentativas de construção de modelos computacionais inspirados na biologia da mente humana e descobertas da neurociência.

Assim como as redes neurais artificiais, criadas inicialmente na década de 50 com as mesmas pretensões (partindo da metodologia de um *perceptron* (ROSENBLATT, 1958)), e que hoje têm revolucionado as ideias inovativas no meio tecnológico, os modelos cognitivos surgiram na década de 70, tendo o mesmo objetivo das redes neurais artificiais de avançar as aplicações tecnológicas. No entanto, os modelos cognitivos têm permitido também estabelecer discussões biológicas mais abrangentes sobre os modelos neurais. Isso só tem sido possível porque esses modelos podem hoje aproveitar mais e melhor os avanços recentes do hardware relacionado, que hoje permite o processamento de dados superior ao de décadas passadas. Apesar disso, os objetivos desta seção não são de incluir nem estimular discussões da área biológica-neural, mas apresentar os principais modelos inspirados cognitivamente do ponto de vista computacional, e que têm estrita relação com este trabalho de dissertação de mestrado. Embora as citações aqui sejam majoritariamente do ponto de vista computacional, sempre que possível será feito relacionamento à inspiração na área da medicina sem, no entanto, desfocar do propósito principal que é o uso desses modelos para reconhecimento de objetos em cenas naturais.

Kotseruba e Tsotsos (2018) estudaram os últimos 40 anos das arquiteturas cognitivas e organizou as informações para que seja possível fazer uma revisão geral da área dos modelos cognitivos. Uma observação importante dentro do conjunto de trabalhos estudados é que, muitas arquiteturas cognitivas foram propostas visando diferentes objetivos. A Figura 1 sumariza as arquiteturas em ordem cronológica (período de utilização), além de diferenciar as arquiteturas simbólicas (verde), emergentes (vermelho) e híbridas (azul). Como pode ser observado, a LIDA está entre as mais recentes arquiteturas propostas, sendo esse um dos principais motivos de a termos escolhido para o estudo.

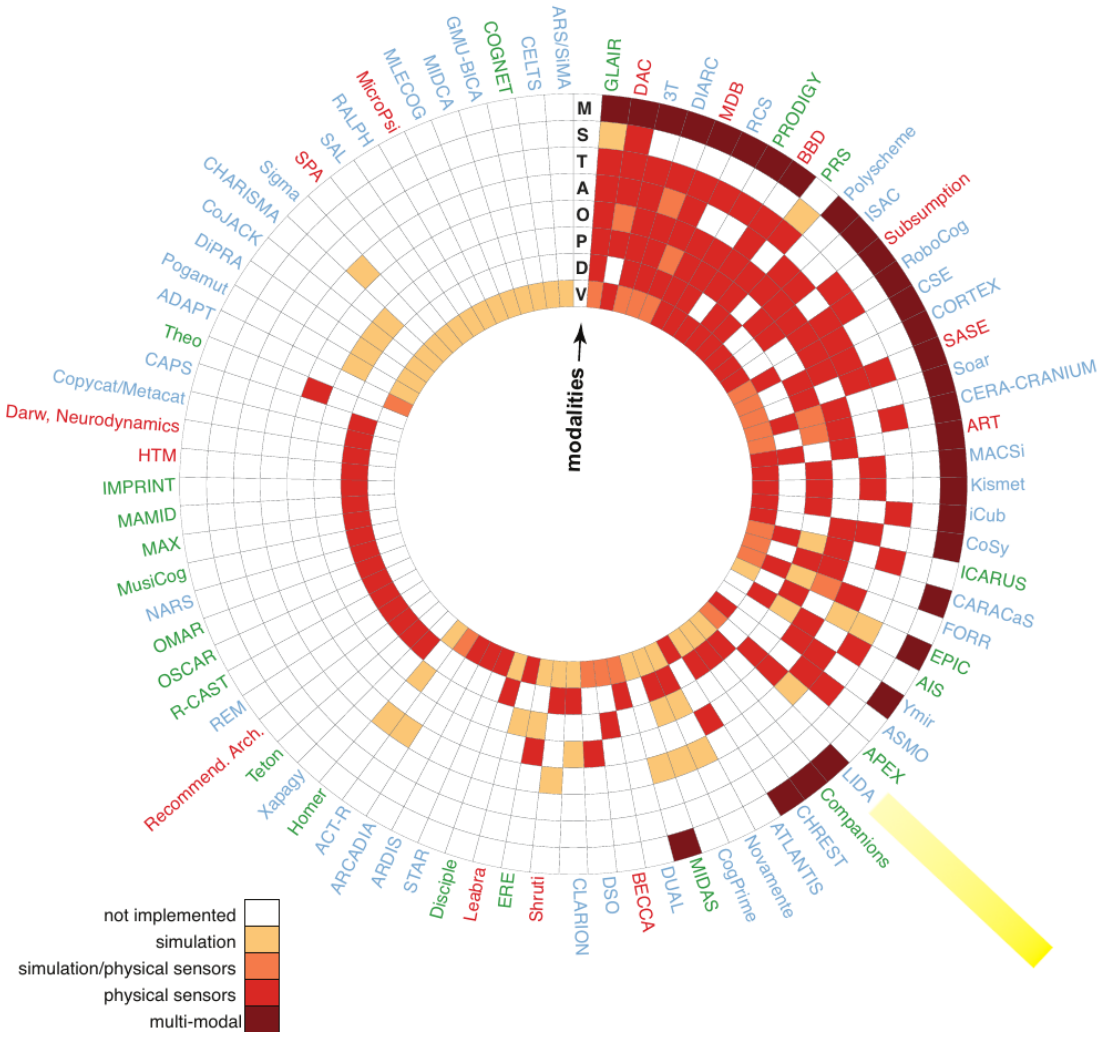
Figura 1 – *Timeline* de 84 arquiteturas cognitivas (simbólicas em verde, emergentes em vermelho e híbridas em azul) em ordem cronológica.



Fonte: Adaptado de Kotseruba e Tsotsos (2018)

O autor ainda estudou as modalidades sensoriais empregadas em cada arquitetura, como ilustra a Figura 2. Foram considerados os sensores visual (V), simbólico (D), propriocepção (P), outros (O), audição (A), tato (T), olfato (S) e os sensores de multi-modalidades (M). Além disso, foram estudados os níveis de implantação dos sensores, partindo dos que foram somente simulados até os que foram fisicamente integrados. Como pode ser observado, a LIDA é capaz de atuar em um ambiente virtual utilizando o sensoriamento visual (como vídeos ou imagens).

Figura 2 – Capacidades sensoriais de cada arquitetura.

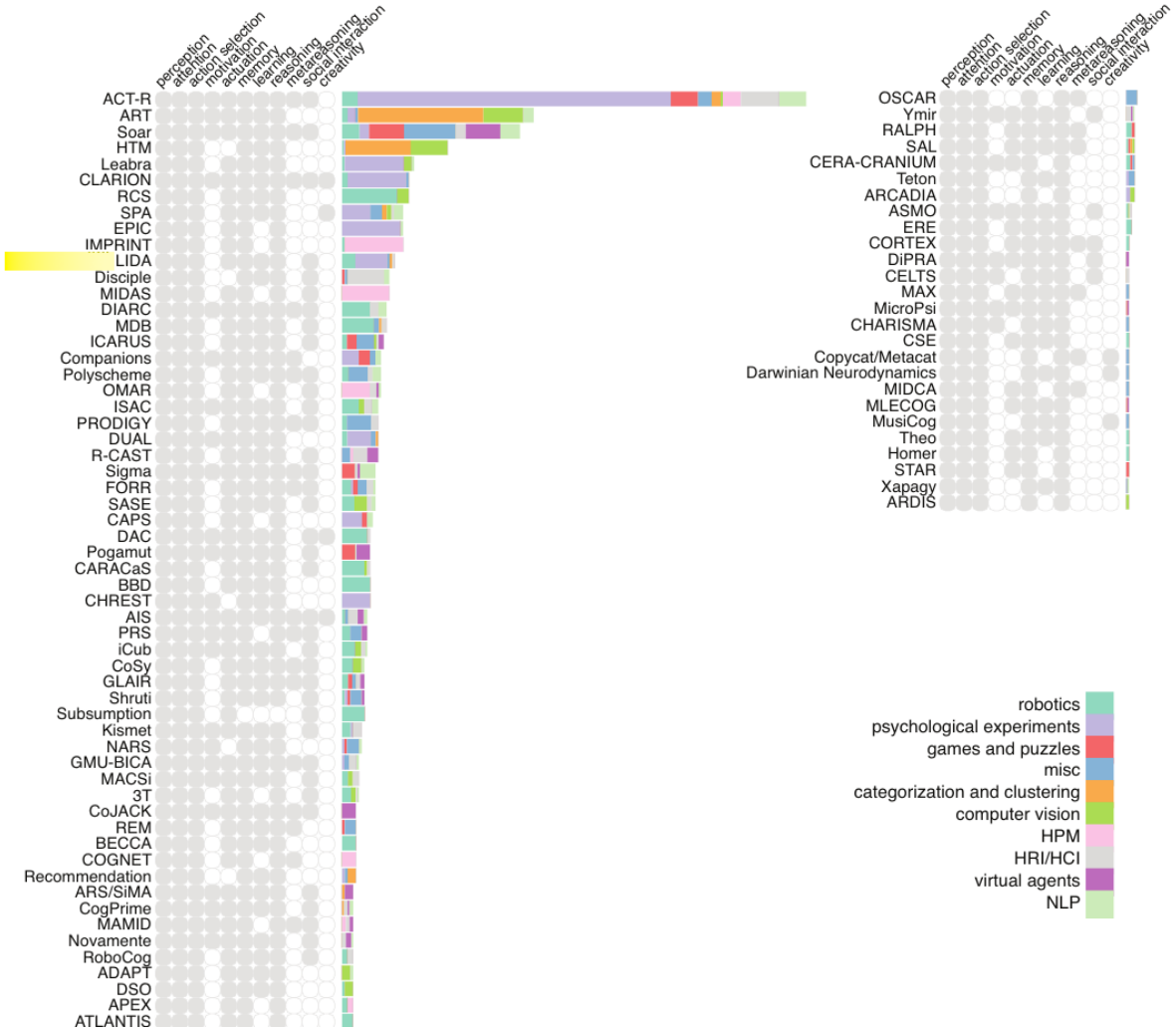


Fonte: Adaptado de Kotseruba e Tsotsos (2018)

A Figura 3 apresenta as aplicações práticas de cada arquitetura cognitiva e a quantidade de aplicações desenvolvidas (comprimento das barras no gráfico). As aplicações são separadas em áreas definidas por Adams et al. (2012) de modo a ilustrar tarefas reais que são desempenhadas pro modelos computacionais. Observa-se que a LIDA, apesar de ser uma das mais recentes arquiteturas propostas, é uma das que mais possui aplicações desenvolvidas. Outro ponto é

o fato da utilização mínima da LIDA em aplicações da área de visão computacional, o que consolida a importância do presente trabalho em trazer tal arquitetura para tratar de problemas relacionados à esta área.

Figura 3 – Aplicações práticas de cada arquitetura cognitiva.



Fonte: Adaptado de Kotseruba e Tsotsos (2018)

2.3 DISCUSSÃO

Com base no que foi visto neste capítulo, é possível observar que a LIDA foi pouco explorada em visão computacional. Em contrapartida, as redes neurais e o SVM possuem uma grande presença na literatura na tarefa de classificação de cenas. Dado isto, a contribuição deste trabalho está em unir estes dois campos de pesquisa, de modo a se beneficiar das qualidades de cada um.

3 CONCEITOS FUNDAMENTAIS

Neste capítulo, serão apresentados os principais conceitos que estarão presentes neste trabalho que, por sua vez, visa unir métodos muito utilizados na literatura (apresentados no capítulo anterior) e a arquitetura cognitiva LIDA.

3.1 DEEP LEARNING

O primeiro conceito abordado é o Deep Learning (redes neurais profundas), com o qual é possível estruturar padrões complexos a partir da união de padrões mais simples (LECUN; BENGIO; HINTON, 2015). O Deep Learning consiste em criar uma rede de elementos processadores distribuídos em camadas escondidas (intermediárias) de uma rede neural. Com isso, o Deep Learning é conceituado como uma variação aprimorada das redes neurais tradicionais.

Nesta seção serão apresentados os modelos de Deep Learning utilizados neste trabalho, incluindo os elementos processadores empregados em cada modelo.

3.1.1 Multilayer Perceptron (MLP)

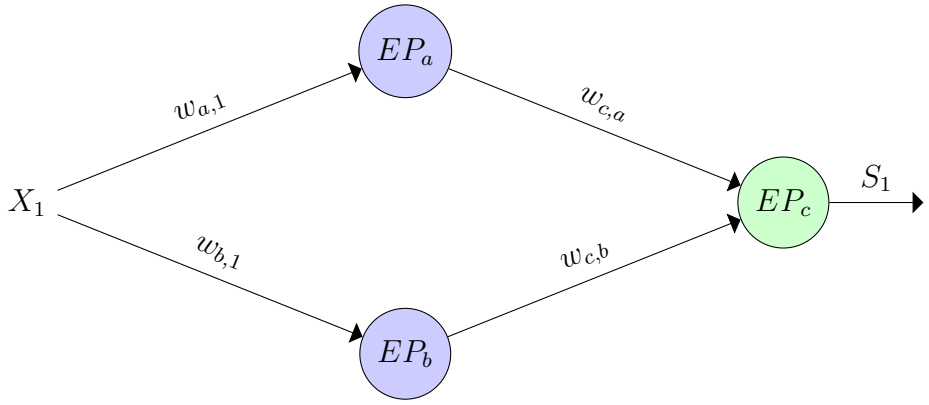
A *Multilayer Perceptron* (MLP), também conhecida como rede neural *feedforward*, é um modelo de Deep Learning que mapeia um conjunto de entradas a valores de saída através de uma função matemática (LECUN; BENGIO; HINTON, 2015). O autor ainda aponta que essa função matemática é uma composição de funções matemáticas mais simples.

3.1.1.1 Inferência

O processo de inferência deste modelo se inicia multiplicando os valores de entrada ($X_{1...n}$ providos dos dados de uma amostra) aos seus pesos ($w_{1...n}$), como é ilustrado na Figura 4. O resultado desta multiplicação é utilizado no processo de inferência dos elementos processadores (representado na Figura 5) da primeira camada escondida.

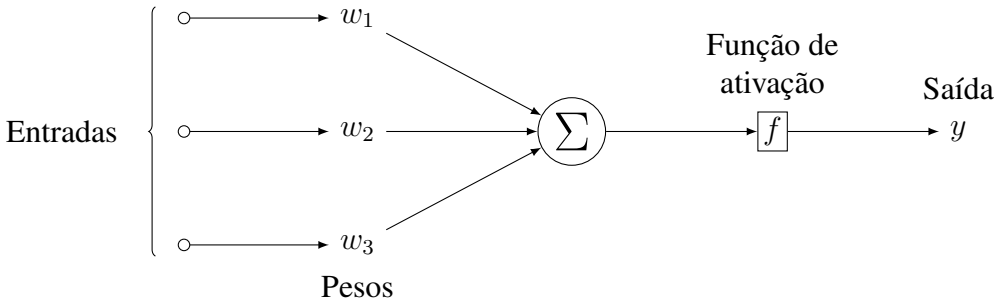
A inferência dos elementos processadores consiste em somar todas as entradas da camada já ponderadas pelos respectivos pesos w e, utilizando o resultado da somatória, mapear um valor através de uma função de ativação. Deste modo, tornando a camada totalmente conectada (*fully connected*) com a camada anterior.

Figura 4 – MLP com 1 entrada, 2 elementos processadores em uma única camada escondida (azul) e 1 elemento processador na camada de saída (verde).



Fonte: Autor

Figura 5 – Detalhe da inferência de um elemento processador da MLP.



Fonte: Autor

Existem diversas funções que podem ser utilizadas para o mapeamento, como a sigmóide (Equação (1)), tangente hiperbólica (Equação (2)), ReLU (Equação (3)), etc. A saída da função de ativação y é então utilizada como entrada para os elementos processadores da próxima camada da MLP, através dos pesos w que as conectam.

$$f(x) = \frac{1}{1 + e^{-x}} = y \tag{1}$$

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = y \tag{2}$$

$$f(x) = max(0,x) = y \tag{3}$$

A camada de saída também irá produzir valores y ($S_n = y_n$) que, além de representarem o resultado da inferência, são utilizados para o treinamento da MLP.

3.1.1.2 Treinamento

O treinamento da MLP consiste em modificar os pesos w (que foram inicializados com valores aleatórios) da rede de modo a aprimorar o resultado das inferências, através do uso de uma função de erro E, exemplificada na Equação (4). Esta função de erro quantifica a distância entre a saída esperada T e a saída obtida S em um treinamento supervisionado.

$$E = (T - S)^2 \tag{4}$$

Utilizando o Gradiente Descendente (que será explicado na Seção 3.3), a variação dos pesos da MLP é obtida através da derivada do erro E em relação aos próprios pesos w, como indica a Equação (5). Após isso, os pesos são atualizados pelo Δw_n, conforme mostra a Equação (6). O parâmetro $0 \geq \alpha \geq 1.0$ existe para minimizar os ajustes locais, permitindo que a MLP aprenda padrões complexos (globais) através das várias amostras de treinamento, também conhecido como taxa de aprendizagem.

$$\Delta w_n = -\frac{\partial E}{\partial w_n} \tag{5}$$

$$w_n = w_n + (\Delta w_n * \alpha) \tag{6}$$

Com um processo iterativo de treinamento das amostras, inclusive utilizando todas as amostras mais de uma vez (este procedimento é conhecido como época), o erro E é decrementado até um critério de parada, que pode ser um valor mínimo de convergência ou o um número de iterações predefinidos. A partir deste ponto, a MLP está pronta para inferir (com mais performance) as amostras de teste, uma vez que ela aprendeu padrões complexos durante o treinamento de seus pesos.

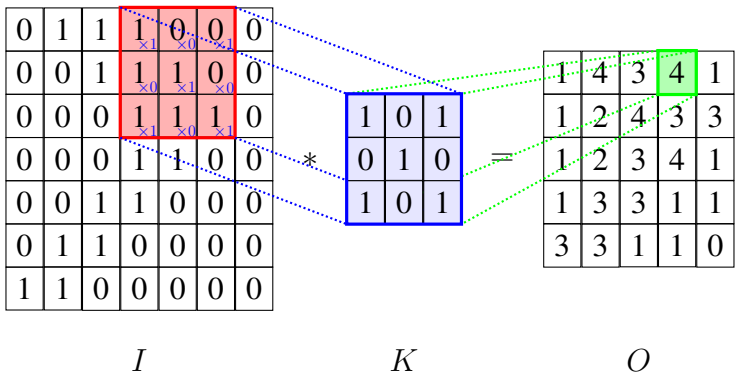
3.1.2 Rede Neural Convolucional

As redes neurais convolucionais (CNN) são uma especialização das redes neurais profundas que processam dados estruturados como matrizes (LECUN; BENGIO; HINTON, 2015). Isto significa que podem processar dados unidimensionais, tais como áudios, ou também bidimensionais, tais como imagens em escala de cinza. Entretanto, através da utilização de tensores, podem processar dados com mais de 2 dimensões.

3.1.2.1 Inferência

O fluxo de inferência das CNNs é semelhante ao da MLP, consistindo que cada elemento processador da rede aplique o processo de convolução, camada-a-camada. Este processo de convolução consiste em aplicar um filtro (*kernel*) K em uma matriz de entrada I; isto é, fazer com que o *kernel* de dimensões $k \times k$ "deslize" sobre a matriz iterativamente e, a cada passo, multiplicando o conteúdo selecionado da matriz pelo próprio *kernel*, como é ilustrado na Figura 6 e na Equação (7). A cada iteração, uma posição da "matriz filtrada" O é calculada, que será novamente filtrada pelos elementos processadores da próxima camada convolucional.

Figura 6 – Processo de convolução.



Fonte: Autor

$$O_{i,j} = \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} I_{a+i,b+j} * K_{a+1,b+1} \tag{7}$$

A camada de saída S de uma rede neural convolucional produz valores y, que é o resultado da aplicação de vários filtros presentes na rede. Este resultado será utilizado na inferência e no treinamento, assim como a MLP (veja a Figura 4). Para isso, pelo menos esta última camada deve ser estruturada igualmente à MLP, com uma soma de entradas e uma função de ativação. Desta forma, as informações filtradas através de várias camadas de convolução são mapeadas por uma função $f(x)$. Além disso, faz com que as camadas convolucionais trabalhem para a extração de *features*, enquanto as camadas *fully connected* (FC) desempenham a classificação. Adicionalmente, é adicionada uma camada *Softmax* na saída, cuja função é somente normalizar os valores y entre 0 e 1.

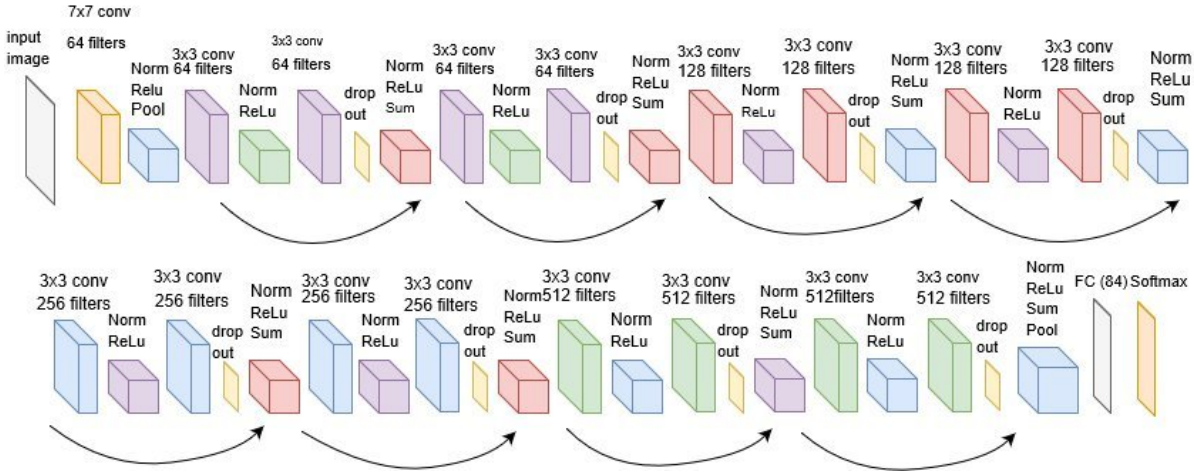
3.1.2.2 Treinamento

O treinamento das redes neurais convolucionais consiste em ajustar os valores do *kernel* K (que foram inicializados com valores aleatórios) utilizando a saída S; isto é, o próprio elemento processador armazena a informação aprendida, ao contrário da MLP que armazena em pesos que conectam os elementos processadores. Assim, cada filtro faz uma operação matemática simples, mas toda a rede faz uma operação complexa. O processo de treinamento, assim como na MLP, ajusta os valores do elemento processador (*kernel*) utilizando o Gradiente Descendente.

Com isso, a motivação deste modelo está na vantagem de trabalhar com dados de maiores dimensões (O'SHEA; NASH, 2015). Isso porque as informações adquiridas no treinamento ficam armazenadas nos *kernels* K que podem possuir menos parâmetros do que a "quantidade pesos" w em uma MLP, por exemplo. Assim, cada *kernel* armazena uma padrão relevante para ser destacado das demais informações. Esta é a razão pela qual esse *kernel* é denominado de filtro.

3.1.2.3 ResNet

Figura 7 – Rede neural Resnet-18.



Fonte: Alif, Ahmed e Hasan (2017)

A *Residual Neural Network* (HE, K. et al., 2016), também conhecida como ResNet, é um modelo de rede neural convolucional que faz uso de informações residuais. A informa-

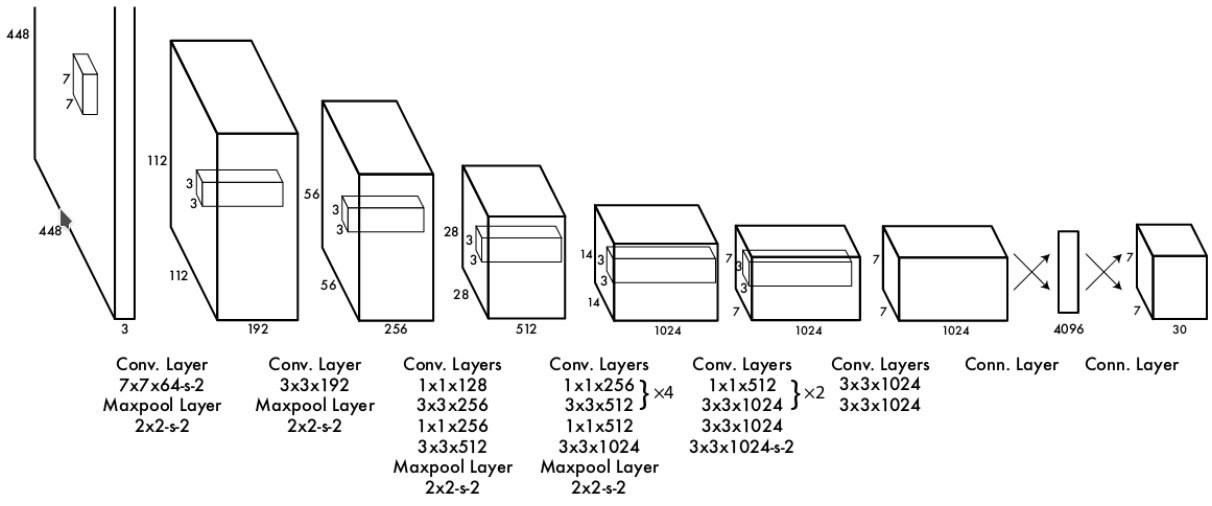
ção residual consiste em concatenar a saída de uma camada convolucional com a de outra. Desta forma permitindo que informações iniciais de convolução sejam prolongadas ao longo da inferência, fazendo com que os filtros das camadas mais profundas tenham mais dados para detectarem padrões.

Além disso, a ResNet conta com mais componentes para aprimorar acurácia da inferência (que serão explicados na Seção 3.1.3): *Batch Normalization* com a função ReLU, *Dropout*, *Pooling* e *Softmax*. A Figura 7 ilustra a estrutura de uma ResNet-18 (18 camadas entre convolucionais e *fully connected*) indicando as concatenações de informações residuais (setas).

3.1.2.4 YOLO

A rede neural convolucional *You Only Look Once*, também conhecida como YOLO (REDMON et al., 2016), é um modelo desenvolvido para detectar objetos; isto é, localizar objetos presentes em uma imagem. Como o autor cita, este modelo é capaz de detectar objetos (determinar *bounding boxes*) juntamente com as distribuição de probabilidades para cada classe de objeto, utilizando somente uma única rede neural, que está ilustrada na Figura 8. Para atender esta característica, a YOLO faz um único processo de regressão para detectar os objetos.

Figura 8 – Rede neural YOLO.



Fonte: Redmon et al. (2016)

A YOLO performa a detecção através da criação de um *grid* $S \times S$ sobreposto à imagem, que por sua vez é utilizado para determinar um mapa de probabilidades de classes em cada unidade do *grid*; como também, possíveis *bounding boxes* (também conhecidos como *object*

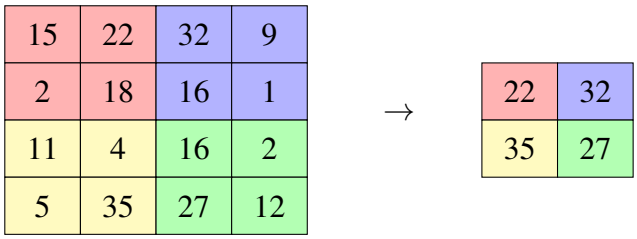
proposals e seus respectivos índices de confiança. A partir dessas duas informações, o modelo localiza os objetos na imagem, como também a distribuição de probabilidades de classes.

3.1.3 Componentes Adicionais

Além das camadas convolucionais, uma CNN pode contar com outros tipos de camadas escondidas de modo a utilizar mais mecanismos para detecção de padrões.

Camada *Pooling*, empregada para reduzir a dimensionalidade de matrizes (ou tensores) em uma rede neural convolucional, como cita Karpathy e Johnson (2019), mantendo as informações mais importantes. Assim, o número de parâmetros é reduzido de modo que o treinamento não conduza o modelo a um estado que fique sensível excessivamente aos ruídos e/ou às repentinas mudanças nas informações de entrada (*overfitting*). A operação de *Pooling* consiste em delimitar regiões da matriz (2×2, 3×3, etc) para então extrair um único valor de cada uma, a fim de compor uma nova matriz utilizando somente um valor por região; a escolha do valor pode ser feita utilizando o valor máximo, médio ou mínimo (*Max Pooling*, *Average Pooling* e *Min Pooling*, respectivamente). A Figura 9 ilustra a operação de *Max Pooling*.

Figura 9 – *Max Pooling* 2×2.



Fonte: Autor

Outro componente importante é o *Dropout* (SRIVASTAVA et al., 2014), que desempenha a função de evitar o efeito *overfitting*. Essa técnica permite uma chance de um elemento processador da camada escondida ser ignorado durante o treinamento, de modo que os valores relacionados a este elemento processador permaneçam inalterados por uma iteração de treinamento. Assim, a rede neural ativa novos caminhos dentro da rede de elementos processadores.

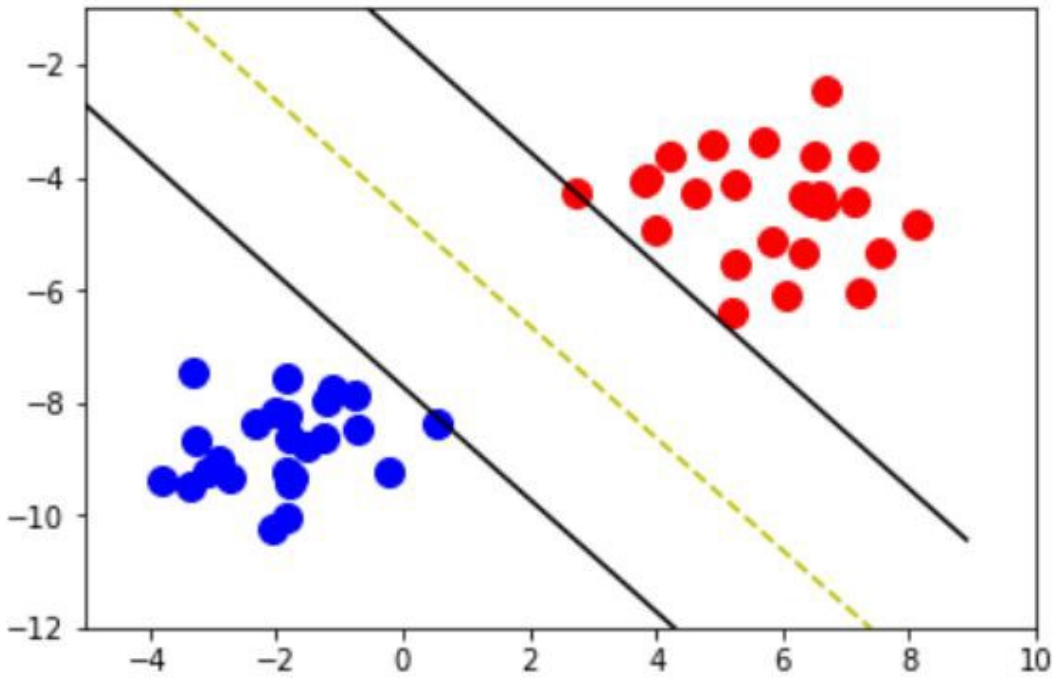
Por fim, existe o *Batch Normalization* (IOFFE; SZEGEDY, 2015) que visa promover um balanceamento nos valores dos parâmetros das camadas escondidas. Isto significa que este componente pode ser aplicado em casos de alguns parâmetros variarem entre 0 e 1 enquanto outros variam entre 0 e 1000. Assim, o processo de *Batch Normalization* normaliza todos os

valores para um mesmo intervalo. O processo consiste de ajustar os parâmetro à uma função de ativação não-linear, como a ReLU, sigmóide ou tangente hiperbólica.

3.2 SUPPORT VECTOR MACHINE (SVM)

Seguindo a lista de métodos amplamente utilizados na literatura que serão empregados neste trabalho, o *Support Vector Machine* (SVM) que é um modelo voltado para a classificação de dados (STITSON et al., 1996; HSU; CHANG; LIN, 2008), será empregado para classificar imagens. O SVM realiza a classificação buscando um hiperplano que separe duas classes de dados, como indica a Figura 10. Além disso, durante a busca por um hiperplano, o SVM usa a margem para com as amostras de dados como critério de avaliação (quanto maior a margem, melhor é a avaliação).

Figura 10 – Classificação com o SVM.



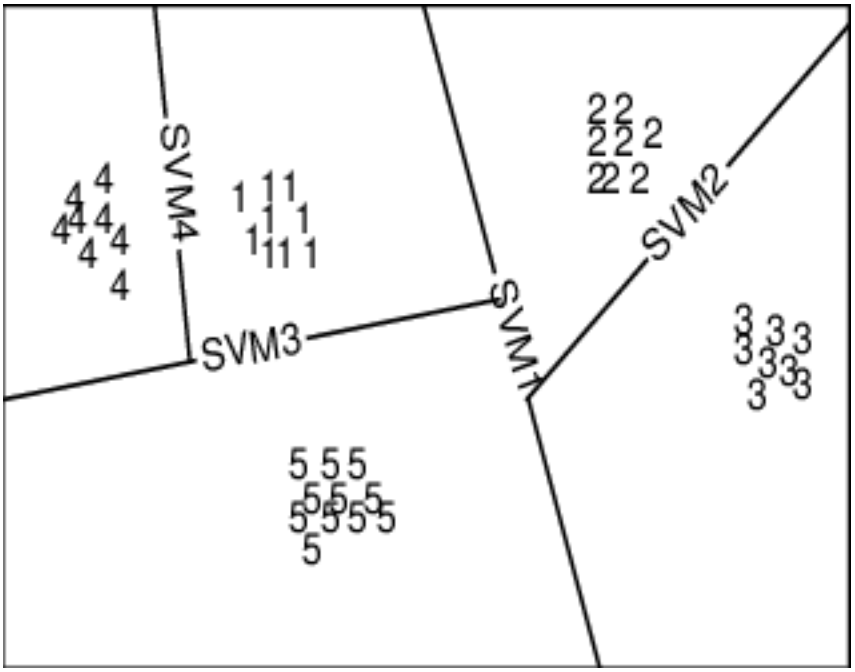
Fonte: Sanjeevi (2017)

3.2.1 Inferência

A inferência do SVM consiste em determinar a posição de uma amostra no espaço em relação ao hiperplano de separação de classes; isto é, determinar em qual das regiões do espaço (que foram definidas pelo hiperplano) a amostra se encontra. Logo, isso significa que o SVM é capaz de trabalhar somente com duas classes.

Para problemas de multiclasses (mais de duas classes) pode ser utilizada a técnica *um versus um* (KNERR; PERSONNAZ; DREYFUS, 1990). Esta técnica consiste em construir $N(N-1)/2$ classificadores SVM dadas N classes de dados, assim cada classificador realiza a inferência com duas classes. A Figura 11 ilustra esta técnica. Deste modo, cada classificador contribui com um voto para uma classe e, consequentemente, a classe mais votada é tida como vencedora da classificação multiclasse.

Figura 11 – Classificação multiclasse com o SVM.



Fonte: Liu et al. (2005)

3.2.2 Treinamento

O treinamento do SVM consiste em determinar o *kernel* descrito na Equação (8), que por sua vez conta com o vetor hiperplano x, a hiper linha de peso w e um *bias* b (SANJEEVI, 2017).

Este *kernel*, denominado linear, tem os parâmetros w e b ajustados pelo Gradiente Descendente durante o treinamento, de modo a buscar a maior distância para com os vetores de suporte.

$$w^T * x + b = 0 \tag{8}$$

Os vetores de suporte são definidos pelas amostras que estão na fronteira entre as duas classes, como indica a Figura 10. Através destas, são definidos os hiperplanos inferior e superior utilizados na determinação da margem (SANJEEVI, 2017).

3.2.3 Kernels

Além do *kernel* linear descrito há pouco, existem outros que são amplamente conhecidos na literatura (HSU; CHANG; LIN, 2008). Dentre eles, os mais conhecidos são: o RBF ($e^{-\gamma\|w-x\|^2}$) que busca classificar os dados com base em elipsóides; e o sigmóide ($sigmoid(\gamma * w^T * x + r)$) que, com base na Equação (1), infere dados utilizando a não-linearidade. Estes *kernels* trazem parâmetros adicionais (γ e r) para serem ajustados pelo Gradiente Descendente.

3.3 GRADIENTE DESCENDENTE

Frequentemente, na computação, modelos matemáticos são utilizados para abstrair comportamentos nas mais diversas áreas (ANDRYCHOWICZ et al., 2016), tais como: física, química, biologia, entre muitas outras. Contudo, esses modelos não são rígidos; isto é, eles apresentam parâmetros para se ajustarem a um determinado conjunto de dados. Na literatura, esse ajuste pode ser feito através de diversas técnicas. Uma técnica bem conhecida é através da otimização.

A otimização é um processo matemático capaz de encontrar mínimos ou máximos em uma função. Por exemplo, sabe-se que quando a primeira derivada de uma função é 0, tem-se o ponto de mínimo, máximo ou de inflexão. Contudo, a medida que estes modelos se tornam mais complexos, com mais variáveis paramétricas, a primeira derivada pode não ser a melhor estratégia. Assim, métodos numéricos foram criados para fazer a mesma tarefa em domínios mais complexos (WITTEN et al., 2016).

Um método bem conhecido na computação é chamado de Gradiente Descendente. Esse método é iterativo e visa minimizar o valor de uma função; isto é, dada uma função qualquer $y = f(x_1, \ldots, x_n)$ deseja-se encontrar os valores de $x_1, \ldots, x_n$ tal que y seja mínimo. Este

método assume que a função f é contínua no intervalo que se deseja otimizar e diferenciável (WITTEN et al., 2016). A ideia geral desse método é efetuar derivadas parciais em relação aos parâmetros da otimização e usar o valor de seus coeficientes angulares, calculados no ponto atual da função, como uma direção para a atualização de cada parâmetro.

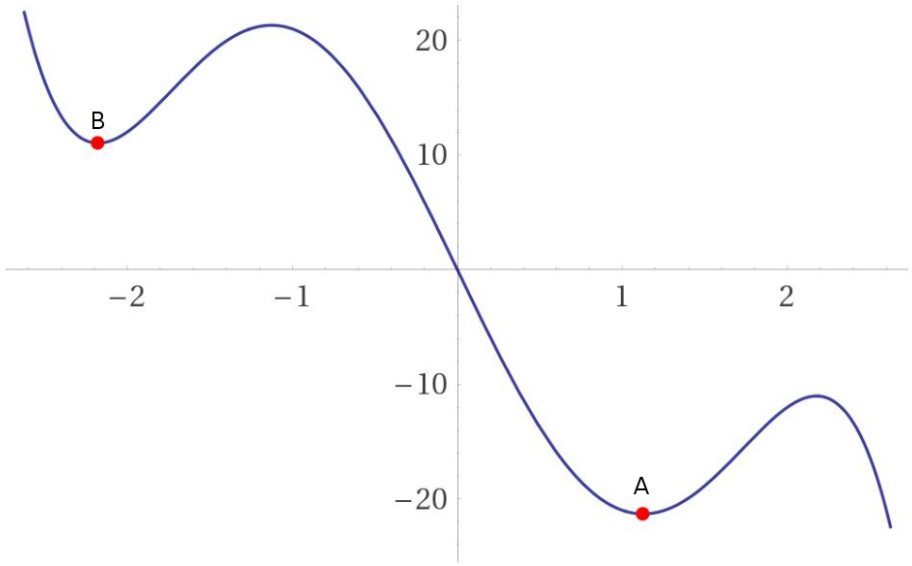
A ideia de seguir a derivada parcial de uma função é utilizada (NARENDRA; PARTHASARATHY, 1991) uma vez que sempre que a derivada de uma função é negativa, a função está em um ponto de declive. Assim, se $\frac{\partial f(x_1,...,x_n)}{\partial x_1} < 0$, então a função encontra-se em um declive a partir do ponto de vista da variável x_1. Dessa forma, é vantajoso seguir a otimização aumentando o valor de x_1, pois espera-se que o valor da função continue caindo. Caso o valor da derivada parcial fosse positivo, a otimização deveria seguir o lado oposto, diminuindo o valor de x_1. Esse processo é executado para cada parâmetro da função (no caso da MLP, cada w é um parâmetro de uma função global da rede).

Apesar dessa técnica parecer ótima, alguns problemas podem surgir durante a otimização. Um destes problemas é que a função que se deseja minimizar pode apresentar diversos mínimos, representados na Figura 12; isto é, cada ponto de f tal que $\frac{\partial f}{\partial x} = 0$ pode não ser o mínimo global da função, que é representado pelo ponto A na figura. Esses pontos são chamados de mínimos locais, como o ponto B exemplifica. Uma técnica utilizada para mitigar esse problema é utilizando uma taxa de aprendizagem. Essa taxa pondera o coeficiente angular da derivada parcial, multiplicando o seu valor por um valor real entre 0 e 1. Assim, se essa taxa for muito baixa, os valores de x sofrerão poucas mudanças e o método deverá iterar mais vezes para alcançar um mínimo. Por sua vez, se a taxa for muito alta, poderá haver menos iterações, porém as otimizações podem perder precisão (WITTEN et al., 2016).

A ideia aqui é criar uma função de erro que associa valores de entrada com valores de saída esperada (HOMOD et al., 2012). Se esta função for minimizada ao longo de todas as amostras, a rede estará "errando menos" e o modelo estará convergindo para o ideal.

Além deste conceito apresentado, existem outras variações deste modelo de otimização, como *Stochastic Gradient Descent* (SGD) e o Adam. O SGD é um modelo muito semelhante ao Gradiente Descendente descrito a pouco, a diferença está no fato do SGD utilizar uma única amostra (ou em alguns casos, um sub-conjunto de amostras) aleatória a cada iteração de otimização, de modo a reduzir o custo computacional da otimização, como cita Ruder (2016). Por outro lado, o Adam (KINGMA; BA, 2014) é um modelo aprimorado do SGD que faz o uso do *momentum* para possibilitar uma adaptação durante a otimização, de modo que os ajustes realizados em iterações passadas seja utilizado para ajustar o valor corrente.

Figura 12 – Mínimos de uma função.



Fonte: Autor

3.4 LIDA

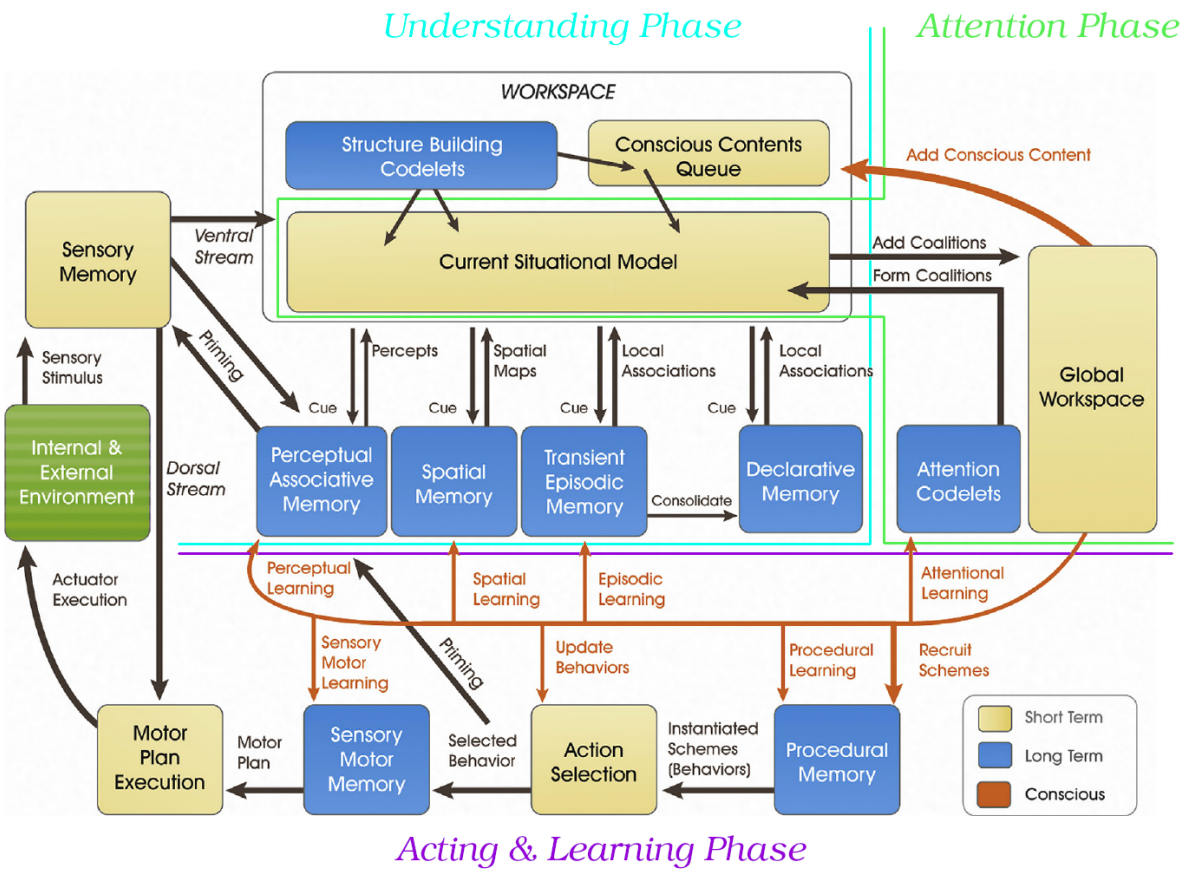
As arquiteturas cognitivas, conforme o que foi apresentado na Seção 2.2, são utilizadas há várias décadas para tratar problemas de diversas áreas. Estas arquiteturas podem ser categorizadas como modelos conceituais, matemáticos e computacionais (FRANKLIN et al., 2016). Entretanto, todas compartilham a mesma finalidade de modelar processos e representações cognitivas.

Como cita Franklin et al. (2016), a maioria das arquiteturas cognitivas modelam somente um tipo de processo cognitivo (atenção, percepção, memória, etc). O autor ainda comenta que, são raras as arquiteturas cognitivas que abrangem diversos processos e representações, que é o caso da LIDA.

A arquitetura cognitiva LIDA é categorizada como conceitual e parcialmente computacional (FRANKLIN et al., 2016). Para estruturar os processos e representações presentes na arquitetura, a LIDA conta com seu ciclo cognitivo, ilustrado na Figura 13, que é a base todos procedimentos de alto nível de cognição, como imaginação, raciocínio e planejamento. A LIDA é fortemente baseada na Teoria de Baars (para mais detalhes, veja (BAARS, Bernard, 1988; BAARS, B.J., 1997; BAARS, B. J., 2002)), como afirma Franklin et al. (2014).

O ciclo cognitivo da LIDA pode ser separado em 3 fases: a fase de compreensão, a fase de atenção e a fase de ação e aprendizagem. Em cada etapa existem vários módulos que desempenham um papel específico no ciclo. Este ciclo é uma representação da cognição humana e,

Figura 13 – Ciclo cognitivo da arquitetura LIDA.



Fonte: Adaptado de Franklin et al. (2016)

consequentemente, é muito complexo; logo, para uma única finalidade como a classificação de cenas, determinados módulos não são utilizados.

3.4.1 Fase de Compreensão

A fase de compreensão é o processo de iniciar o ciclo cognitivo da LIDA, começando com a identificação de estímulos do ambiente. Estes estímulos são enviados para a memória sensorial que atua como um *buffer*. As informações armazenadas na Memória Sensorial (*Sensory Memory*) são classificadas como "baixo-nível", uma vez que são os dados brutos coletados do ambiente.

As informações da memória sensorial são então enviadas para o módulo *Perceptual Associative Memory* (PAM), para que sejam utilizadas na identificação de informações de "alto-nível", como por exemplo os objetos em uma cena (FRANKLIN et al., 2014). Neste ponto,

vale citar o módulo EPAM proposto por Madl et al. (2016). Este módulo, como explicado na Seção 2.1, faz o mapeamento entre as informações de baixo-nível e alto-nível utilizando um rede neural convolucional (CNN). Com as informações de alto-nível identificadas (percepções), estas são então enviadas ao *Workspace*.

O *Workspace* é um módulo da LIDA que está na intersecção entre a fase de compreensão e atenção, pois contém sub-módulos voltados especificamente para cada fase. Dentre estes módulos, está o *Current Situational Model* (CSM), que desempenha a função de consolidar as informações ao estado atual da LIDA; ou seja, o CSM armazena somente as informações relevantes para uma tarefa específica que a LIDA está desempenhando no estado atual, além de ser a ponte de comunicação com a fase de atenção. Outro sub-módulo presente no *Workspace* é o *Structure Building Codelets* (SBC), que é formado por um conjunto de processadores (*codelets*[1]) que realizam as alterações necessárias no CSM. Por fim, o sub-módulo *Conscious Contents Queue* é um sistema de memória de muito curto prazo (em humanos, as informações são armazenadas por cerca de três segundos, como cita Franklin et al. (2016)); as informações neste sub-módulo provém de dados conscientes do *Global Workspace*, com o intuito de relacionar as ações que foram tomadas "muito recentemente" com o estado atual do modelo.

Outro módulo presente na LIDA é a chamada Memória Espacial (*Spatial Memory*) que é responsável por representar as posições dos elementos importantes para o modelo; isto é, este módulo gerencia as informações espaciais (ex: cima, baixo, esquerda, etc) relacionadas ao ambiente do qual a LIDA está inserida.

A Memória Declarativa (*Declarative Memory*) é modulo que gerencia as informações de longo-prazo envolvidas na LIDA (em humanos, seria o conhecimento de toda uma vida como cita Franklin et al. (2016)). Estas informações provém da *Transient Episodic Memory*, que será explicada adiante. Estas informações são estruturadas como dois tipos na Memória Declarativa, as informações autobiográficas que possuem o conhecimento do tipo "quando" e "onde"; e as informações semânticas do tipo "o que", que são representadas como fatos ou regras.

A *Transient Episodic Memory* é o módulo responsável por coletar o conhecimento declarativo da LIDA. Este conhecimento é composto de informações dos tipos "o que", "quando" e "onde". Tais informações são vistas como episódios identificados pela LIDA em um intervalo de tempo (em humanos, seria equivalente ao conhecimento de algumas horas ou um dia como pontua Franklin et al. (2016)). A cada intervalo, os episódios mais relevantes são consolidados

[1] *Codelets* são processadores que atuam sobre as informações da LIDA, de modo a identificar um conhecimento.

na Memória Declarativa (semelhante ao sono REM em humanos), para que sejam armazenados por um prazo mais longo na arquitetura.

3.4.2 Fase de Atenção

Basicamente, a fase de atenção é composta por dois módulos. O primeiro é o módulo dos *codelets* de atenção, que é composto por processadores (assim como o SBC descrito na Seção 3.4.1) que buscam coalizões (padrões) no CSM, os quais o modelo deve "prestar atenção".

O segundo é o *Global Workspace*, que deve armazenar informações sobre o "objetivo" do modelo; isto é, funciona como o papel da consciência segundo a Teoria de Baars.

Nesta fase destaca-se o mecanismo de atenção. Esse mecanismo é utilizado para filtrar informações que são úteis da fase de compreensão. Isso é feito armazenando padrões do CSM no *Global Workspace*, identificados como relevantes pelos *codelets* de atenção.

3.4.3 Fase de Ação e Treinamento

A fase de ação e treinamento possibilita à LIDA decidir "o que fazer a seguir". Uma das tarefas realizadas nesta fase é o *broadcast* de treinamento da LIDA a partir das informações do *Global Workspace*, como está ilustrado na Figura 13.

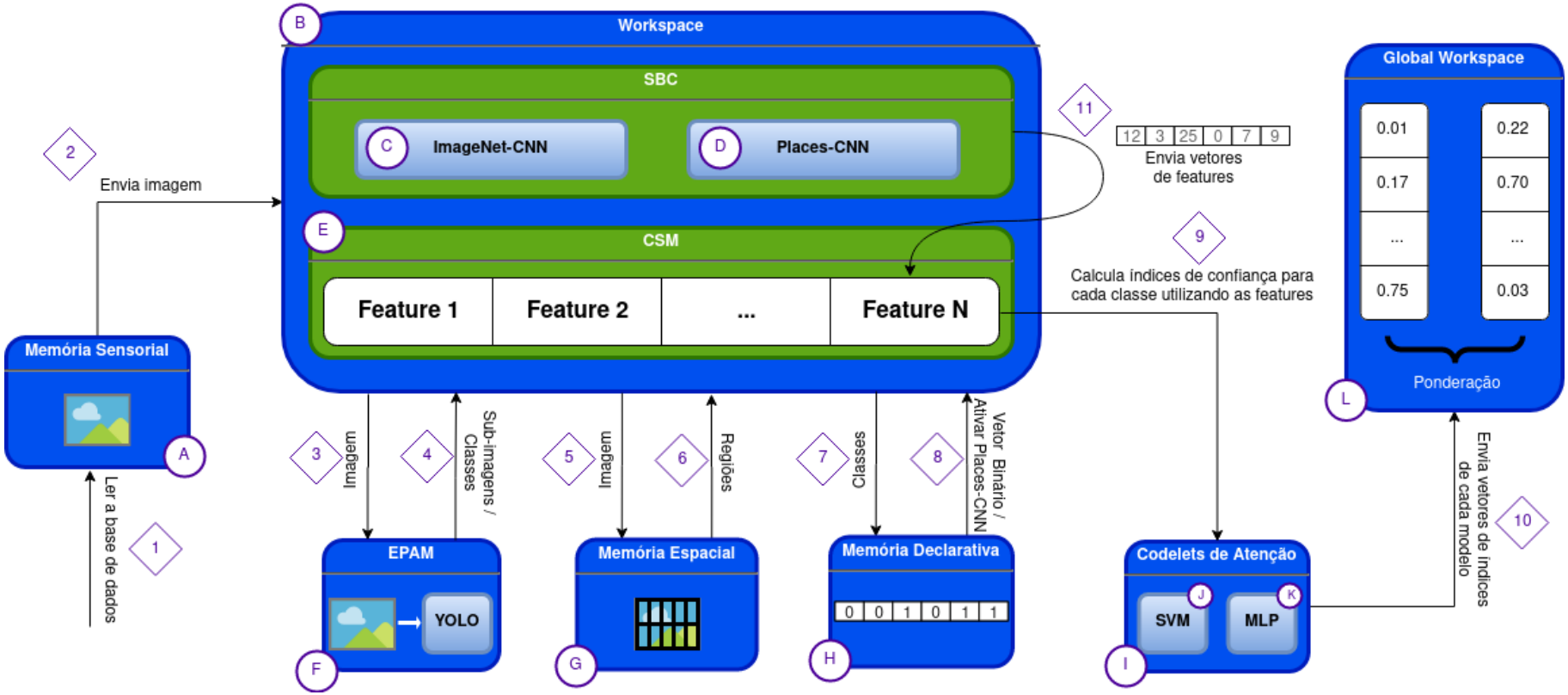
O primeiro módulo abordado nesta fase é a Memória Procedural, que desempenha a função de indicar esquemas de ação. Estes esquemas são planos de ação para cada contexto aprendido pela LIDA.

O módulo de seleção de ação recebe os esquemas indicados para o contexto em que a LIDA se encontra. Dentre estes esquemas, o módulo irá selecionar o mais adequado para ser executado. Além disso, este módulo se comunica com o PAM através de um *priming* para relacionar o esquema que foi selecionado com as informações que estão sendo recebidas dos estímulos.

A *Sensory Motor Memory* (SMS) desenvolve um plano motor para executar o esquema selecionado. Neste módulo, são considerados as atuações para fazer alguma interferência no ambiente em que a LIDA está inserida (por exemplo, quais membros movimentar para subir uma escada, no caso de um robô humanoide). Com este plano, o módulo *Motor Plan Execution* irá realizar a interferência, já considerando os estímulos que a LIDA recebe na Memória Sensorial.

4 METODOLOGIA

Figura 14 – Modelo proposto baseado na arquitetura LIDA (módulos/*codelets* indicados por letras e as conexões por números).



Fonte: Autor

O modelo proposto neste trabalho é estruturado a partir dos conceitos apresentados no capítulo anterior, com o objetivo de unir tais conceitos para realizar a classificação de cenas em imagens. O modelo proposto[1] está ilustrado na Figura 14; e neste capítulo, o mesmo será apresentado através da explicação de cada um de seus módulos.

As seções a seguir descrevem com maiores detalhes as etapas da do modelo proposto.

4.1 BASES DE DADOS

Para avaliar o modelo proposto, as bases de dados MIT Indoor 67 (QUATTONI; TORRALBA, 2009) e SUN 397 (XIAO et al., 2010) serão utilizadas para treino e teste, assim como foram utilizadas nos trabalhos descritos na Tabela 2.

A base MIT Indoor 67 possui 5360 imagens coloridas (3 canais) pré-definidas para treino e 1340 imagens para teste, distribuídas em 67 classes de ambientes internos.

Por outro lado, a base SUN 397 conta com 130.519 imagens coloridas com mais de 100 imagens em cada uma de suas 397 classes (contendo ambientes internos e externos). Assim, utiliza-se a estratégia proposta por Xiao et al. (2010), onde são definidas 10 configurações distintas para treino e teste. Em cada uma destas configurações estão definidas 50 imagens dedicadas ao treino e outras 50 para teste em cada classe.

4.2 MEMÓRIA SENSORIAL

De acordo com o modelo proposto, a Memória Sensorial (módulo A da Figura 14) recebe a informação (imagem) do ambiente, que neste caso é a base de dados (conexão 1). Deste modo, a Memória Sensorial desempenha a função de ler serialmente a base de dados. Após isso, a imagem é enviada ao o *Workspace* (conexão 2), para ser armazenada e acessada durante as inferências posteriormente.

4.3 WORKSPACE

O *Workspace* (módulo B) exerce o papel de processar a imagem através dos *Structure Building Codelets* (SBC), além de compilar, intermediar e estruturar as informações contidas

[1] A implementação completa do modelo proposto está disponível em: https://github.com/HenriqueXG/msc

nas memórias: EPAM, Memória Espacial e Declarativa (módulos F,H e G, respectivamente na Figura 14).

4.3.1 Structure Building Codelets (SBC)

Este módulo é composto por dois *codelets*, mais especificamente duas redes neurais convolucionais, que extraem as *features*[2] da imagem e as enviam para o CSM (conexão 11), que será explicado mais adiante. O propósito desta abordagem é aproveitar a capacidade de identificação de *features* já comprovada na literatura (ver Seção 2.1) através deste tipo de rede neural.

A primeira rede neural utilizada neste módulo é uma Resnet-18 (explicada na Seção 3.1.2.3, *codelet* C) pré-treinada na base de dados ImageNet (DENG et al., 2009) que, como visto em trabalhos relacionados, ficou conhecida como ImageNet-CNN. Esta base de dados é utilizada para a classificação de objetos contando com e 1,2 milhão de imagens com a supervisão de 1000 classes de objetos, Assim, a ImageNet-CNN possui filtros treinados para detectar padrões locais (objetos). Para obter as *features* de objetos desta rede neural, extrai-se os valores de saída da última camada de *Pooling*, a camada *Average Pooling*, reestruturada como um vetor.

Por sua vez, a segunda rede neural (*codelet* D) trata-se de uma Places-CNN, que é uma Resnet-50 pré-treinada na base de dados Places-365 (ZHOU et al., 2014). Esta base de dados é focada na classificação de cenas num total de 1,8 milhão de imagens dispostas em 365 classes de cenas. Deste modo, a Places-CNN realiza a extração *features* globais da imagem de entrada. Assim como na ImageNet-CNN, esta rede neural usa a camada *Average Pooling* para extrair *features* utilizando o *transfer learning* (utilização do conhecimento de uma base de dados para avaliar outro conjunto de dados (TORREY; SHAVLIK, 2010)).

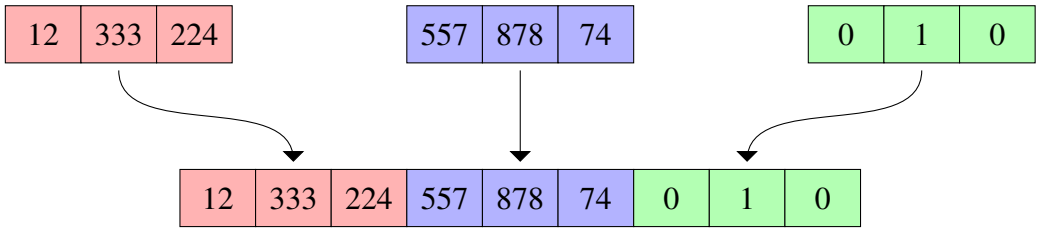
4.3.2 Current Situational Model (CSM)

O CSM (módulo E) possui a tarefa de armazenar um vetor de *features* que é o resultado da concatenação de vetores individuais provenientes de cada módulo do modelo, seja dos *codelets* ou das memórias; esta concatenação está ilustrada na Figura 15. Deste modo, o vetor do

[2] A implementação de img2vec utilizada como base está disponível em: https://github.com/christiansafka/img2vec

CSM é o centralizador das informações que são utilizadas no treinamento e na inferência do modelo.

Figura 15 – Exemplo genérico de concatenação de *features* provenientes de 3 módulos diferentes (vermelho, azul e verde).



Fonte: Autor

A ideia deste módulo é consolidar informações provenientes de diversas fontes (as memórias e os *codelets* do SBC), de forma que seja possível descrever o "estado atual do modelo"; isto é a inferência de uma imagem. Com isso, é possível afirmar que o CSM é a *bag of features* do modelo proposto.

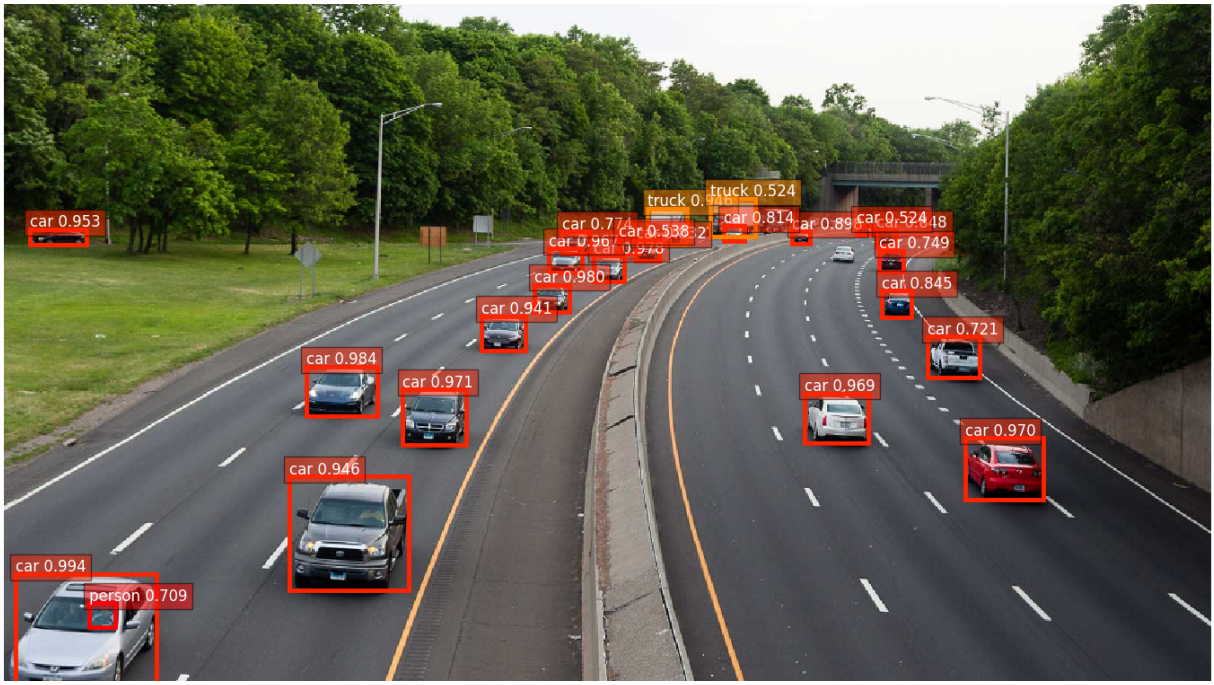
4.4 EXTENDED PERCEPTUAL ASSOCIATIVE MEMORY (EPAM)

Partindo do modelo apresentado por Madl et al. (2016), o EPAM proposto neste trabalho (módulo F) também faz uso de uma rede neural convolucional para processar as imagens. Porém, no modelo proposto neste trabalho é a YOLO[3] (explicada na Seção 3.1.2.4), cuja função é a detecção de objetos (localizar regiões na imagem). A YOLO aqui utilizada é pré-treinada na base de dados Microsoft COCO (LIN et al., 2014) que possui 80 classes de objetos e mais de 300 mil imagens.

Como ilustra a Figura 16, a YOLO detecta objetos na imagem (conexão 3) e os identifica com *bounding boxes*. Um adendo, a YOLO só identifica *bounding boxes* que possuem mais de 50% de confiança (*threshold*) de acerto, de acordo com a implementação de Tong He et al. (2018).

[3] A implementação utilizada desta rede neural está disponível na biblioteca GluonCV (HE, T. et al., 2018).

Figura 16 – *Bounding boxes* detectados associados à uma classe.



Fonte: Autor

Figura 17 – Sub-imagem referente à Figura 16.

Fonte: Autor

Após a detecção, a região da imagem contida em cada *bounding box* é extraída como uma sub-imagem, como mostra a Figura 17. Assumimos aqui a seguinte notação. A imagem fornecida à EPAM é denominada $I \in N \times N$. Uma sub-imagem detectada pelo YOLO é denotada por $I_i, i \in \{1,2,3,...\}$, sendo $I_i \cap I_j, i \neq j$.

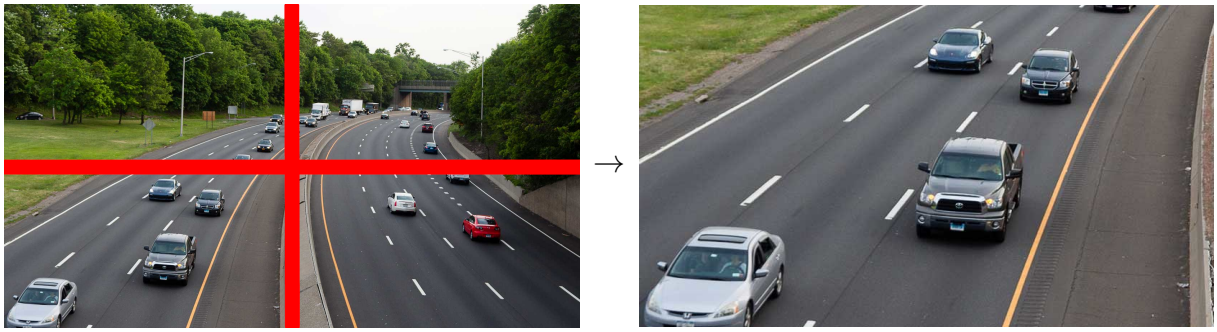
Cada sub-imagem I_i é enviada à ImageNet-CNN (no SBC, através da conexão 4) para extração de *features*. Logo, para cada I_i gera-se um vetor de *features*. Deste modo, para uma determinada imagem que possui várias sub-imagens e, consequentemente, vários vetores de *features*, é feita a soma vetorial normalizada (min-max) de todos esses vetores; de modo a ter-se um só vetor por imagem. Este vetor normalizado é então enviado ao CSM para a concatenação. Assim, através deste vetor, o CSM possui as informações dos objetos que co-ocorreram na mesma imagem, permitindo que seja possível através do CSM descrever a cena a partir de seus objetos.

Além disso, as classes de objetos encontradas na imagem também são enviadas à Memória Declarativa (via *Workspace*) para serem utilizadas posteriormente na construção de um conhecimento declarativo (conexão 4).

4.5 MEMÓRIA ESPACIAL

O módulo espacial (módulo G) deste trabalho utiliza parte da ideia proposta por Zhen e Zhang (2018). Especificamente neste trabalho, a imagem de entrada (conexão 5) é dividida em regiões por um *grid* 2×2 e um 3×3. A Figura 18 destaca essa divisão utilizando um *grid* 2×2.

Figura 18 – Exemplo de uma imagem dividida por um *grid* (à esq.) e uma de suas regiões em destaque (à dir.).



Fonte: Autor

Com isso, cada uma das regiões é enviada à ImageNet-CNN para a extração de *features* (conexão 6) e, logo em seguida, cada vetor gerado é enviado para o CSM. Deste modo, a Memória Espacial traz a habilidade de proporcionalidade de escala para o modelo proposto, semelhantemente ao trabalho de Zhen e Zhang (2018). Além disso, como os vetores gerados são enviados sempre na mesma sequência para o CSM (como por exemplo esquerda acima,

direita acima, esquerda abaixo e direita abaixo), o mesmo possui implicitamente as informações de posições relativas dos objetos; por exemplo: as informações de que o Sol geralmente aparece na parte superior de uma imagem, enquanto o mar aparece na região inferior.

4.6 MEMÓRIA DECLARATIVA

A Memória Declarativa proposta neste trabalho (módulo H) consiste em utilizar as informações providas pelo EPAM, como também extração de *features* através da Places-CNN. Isso é feito de modo a consolidar informações para descrever a cena com informações globais (provenientes da Places-CNN) e explicitamente locais através de um vetor binário, que será explicado adiante.

Como estratégia proposta, as classes recebidas do EPAM (conexão 7) são utilizadas para criar um vetor binário, o qual possui 80 posições (devido às 80 classes existentes na base Microsoft COCO utilizada no EPAM). A Figura 19 ilustra um exemplo de um vetor binário de 3 classes, no qual somente objetos da classe 2 foram detectados no EPAM. Assim, as classes detectadas assumem valor 1 e as demais 0. Após isso, o vetor binário é enviado para concatenação no CSM (conexão 8).

Figura 19 – Exemplo de um vetor binário.

Classe 1	Classe 2	Classe 3
0	1	0

Fonte: Autor

Para descrever a cena de forma global, a Memória Declarativa utiliza a Places-CNN para extrair *features* da imagem de entrada (conexão 8). Como a base de dados Places-365 é uma base com a supervisão de cenas, a rede neural treinada possui filtros ajustados para detectar padrões globais (conhecimento declarativo global). Uma vez que é extraída as *features* da imagem de entrada, o vetor é enviado ao CSM.

4.7 CODELETS DE ATENÇÃO

Nesta seção serão apresentados os *codelets* (módulo I) que classificam as *features* presentes no CSM. Como as informações (*features*) identificadas pertencem à uma imagem por

vez, ao final do processamento dos *codelets* de cada imagem da base de dados, o vetor do CSM é re-inicializado para receber as *features* da próxima imagem; ou seja, é construída uma *bag of features* para cada imagem da base de dados.

4.7.1 SVM

O primeiro *codelet* utilizado é o SVM (*codelet* J), conforme explicado na Seção 3.2, classifica vetores de entrada baseado nos vetores de suporte aprendidos. O *kernel* utilizado neste trabalho é uma função de base radial (RBF), partindo da hipótese de que as *features* podem ser agrupadas por elipsoides, e será feita a comparação com outros *kernels* nos experimentos.

Durante o treinamento do modelo proposto, o SVM utiliza o vetor do CSM como amostra de treinamento. Entretanto, no momento do teste, utiliza o vetor do CSM como amostra.

A saída do SVM, dada uma imagem de entrada I, é um valor de confiança para cada classe possível para I. Essa ideia foi proposta para o LIDA em (FRANKLIN et al., 2016). O esquema da Figura 20 mostra essa ideia. Uma vez criado, o vetor de índices é enviado ao *Global Workspace* (conexão 10).

Figura 20 – Exemplo do funcionamento do SVM durante o teste do modelo.

Fonte: Autor

4.7.2 MLP

O papel da MLP (*Multilayer Perceptron*, *codelet*, bloco K), apresentado na Seção 3.1.1, é semelhante o SVM, explicado anteriormente. A MLP conta com inúmeros parâmetros que podem ser otimizados de modo aumentar a taxa de acerto do modelo proposto em si: quantidade de neurônios na camada escondida, função de ativação, etc. Tais parâmetros são ajustados empiricamente através dos resultados de experimentos que serão descritos no Capítulo 5, de modo a aumentar a performance de inferência da MLP.

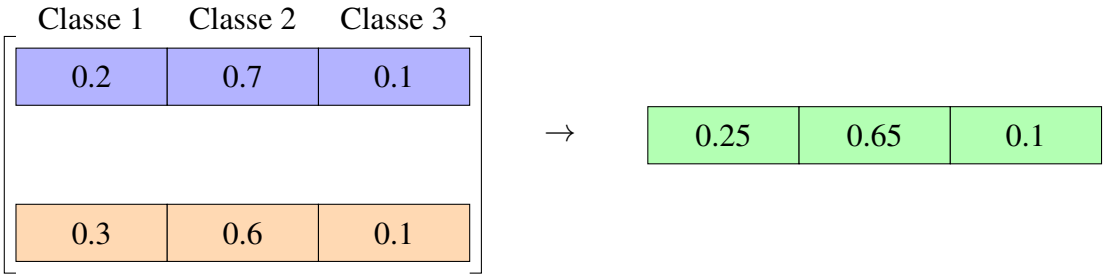
A MLP, assim como o SVM, produz um vetor de índices de confiança de cada imagem (conexão 9) que será enviado ao *Global Workspace* (conexão 10). Um ponto a esclarecer é que,

durante o treinamento, o MLP encerra o ciclo cognitivo proposto (baseado na LIDA); isto é, assim como o SVM, a MLP não envia os vetores de índices de confiança ao *Global Workspace* durante o treinamento, pois ambos *codelets* ainda estão sendo ajustados para posteriormente performar as inferências.

4.8 GLOBAL WORKSPACE

O *Global Workspace* (módulo L) recebe os vetores de índices de cada *codelet de atenção* e une as informações de modo a formar um único vetor por amostra. Para essa finalidade, o *Global Workspace* usa uma variável de ponderação α para unir os valores providos pelo SVM e pelo MLP classe-a-classe. A Figura 21 exemplifica essa ponderação utilizando $\alpha = 0.5$.

Figura 21 – Exemplo de ponderação entre um vetor provido do SVM (azul), MLP (Laranja) gerando o vetor ponderado (verde) para uma determinada imagem.



Fonte: Autor

Uma vez que o *Global Workspace* calcula o vetor ponderado, o mesmo identifica a classe que possui o índice mais alto e, consequentemente, atribui essa classe à imagem em questão (amostra). Quando todas as imagens de teste da base de dados passam por este processo de inferência, o *Global Workspace* verifica as classificações juntamente com a supervisão da base de dados e, por fim, calcula a taxa de acerto do modelo na base utilizada.

5 RESULTADOS

A partir do modelo que foi apresentado na Metodologia, este capítulo apresentará os resultados dos experimentos aos quais foi submetido. Os resultados do modelo proposto são comparados aos de Cheng et al. (2018) e Zhao e Larson (2018), indicados na Tabela 2.

A experimentação do modelo foi executada de forma que as parametrizações foram ajustadas através dos experimentos utilizando a base MIT Indoor 67, apresentados ao decorrer deste capítulo, devido ao número reduzido de amostras em comparação com a base de dados SUN 397. Os tópicos de discussão sobre os experimentos serão apresentados no Capítulo 6.

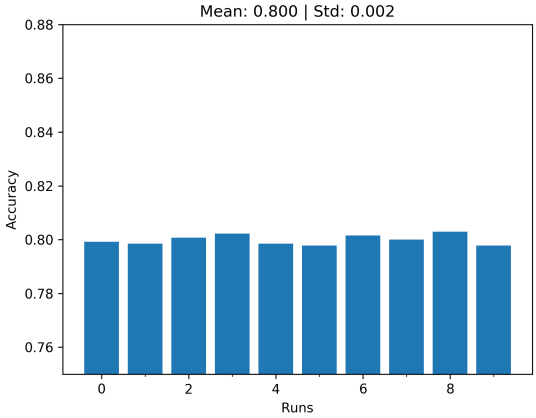
A estratégia de experimentação tem como princípio: a otimização do modelo baseado em LIDA através do aprimoramento das técnicas utilizadas em diversos módulos da LIDA. Além disso, avaliar também a contribuição de cada módulo para a cognição do modelo proposto e o comportamento do modelo cognitivo proposto nos diferentes cenários das bases dados utilizadas. Quanto aos parâmetros utilizados e aprimorados durante a experimentação deste trabalho, os tópicos a seguir identificam cada parâmetro e seus respectivos valores iniciais:

a) *Kernel* do SVM: RBF;
b) Função de ativação da MLP: ReLU;
c) Técnica de otimização da MLP: Adam;
d) Quantidade de elementos processadores da camada escondida (MLP): 1000;
e) *Threshold* da YOLO: 0,5

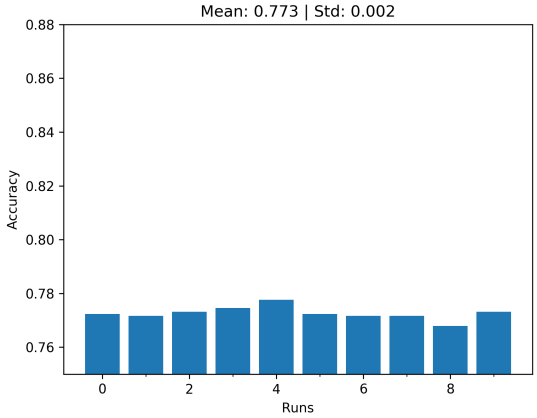
5.1 COMPARAÇÃO ENTRE KERNELS DO SVM

De modo a comparar a utilização proposta de *kernel* RBF para o SVM, foram testados também os *kernels* sigmóide e linear, como ilustra a Figura 22 ao longo de 10 iterações de teste; por deste experimento, foi possível comparar a parametrização do SVM proposta com as demais utilizadas na literatura. Neste experimento somente o SVM foi considerado na classificação $\alpha = 1$.

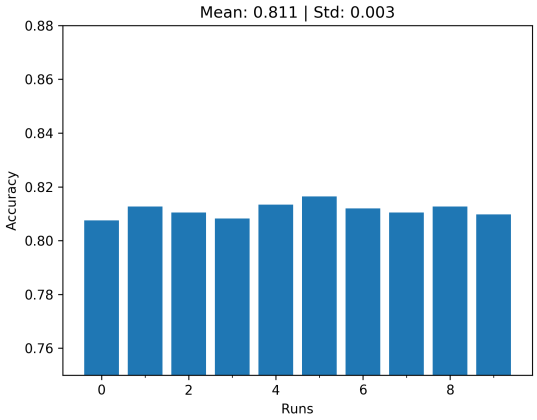
Figura 22 – Experimento para comparar os resultados na base MIT Indoor 67 com os *kernels*: RBF (a), sigmóide (b) e linear (c).



(a) RBF



(b) sigmóide



(c) linear

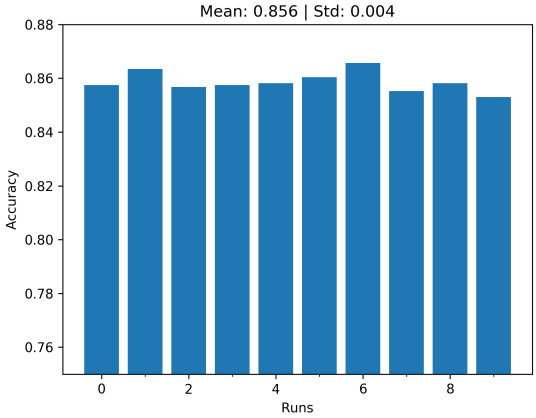
Fonte: Autor

A partir da Figura 22, pode-se afirmar que nenhum *kernel* se destacou com uma significativa vantagem, ainda sim o *kernel* linear obteve a melhor performance com uma média de 81,1% de acurácia ($\pm$ 0,3%) dentre as 10 iterações, seguido do RBF com 80% ($\pm$ 0,2% de desvio padrão) e, por fim, o sigmóide com 77,3% ($\pm$ 0,2%).

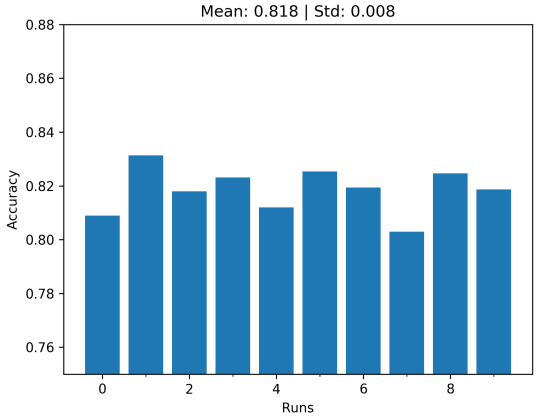
5.2 COMPARAÇÃO ENTRE FUNÇÕES DE ATIVAÇÃO DA MLP

De modo a iniciar a parametrização empiricamente da MLP no modelo proposto, foram avaliadas as funções de ativação sigmóide, ReLU e tangente hiperbólica, como mostra a Figura 23. Neste experimento foram executadas 10 iterações de teste com $\alpha = 0$ (somente MLP).

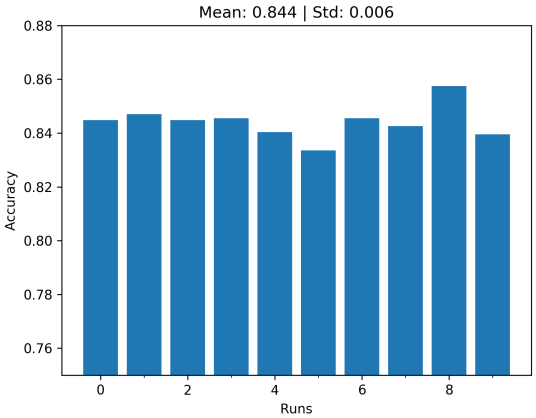
Figura 23 – Experimento para comparar os resultados na base MIT Indoor 67 com as funções de ativação: sigmóide (a), ReLU (b) e tangente hiperbólica (c).



(a) sigmóide



(b) ReLU



(c) tangente hiperbólica

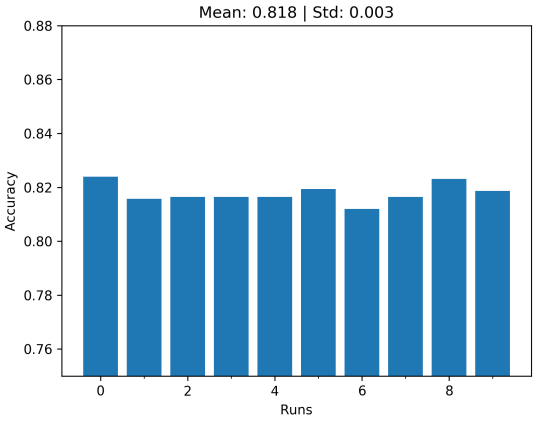
Fonte: Autor

Com os resultados da Figura 23, pode-se afirmar que a função sigmóide alcançou o melhor resultado com uma média de 85,6% (± 0,4%), seguida da tangente hiperbólica com 84,4% (± 0,6%) e, por último, a ReLU com 81,8% (± 0,8%).

5.3 ADAM VS SGD

Uma vez que a sigmóide foi definida como função de ativação por ter o melhor desempenho no experimento anterior, foi identificado empiricamente, a melhor técnica de otimização dentre Adam e SGD (Figuras 23a e 24 respectivamente).

Figura 24 – Performance do modelo proposto na base MIT Indoor 67 utilizando a a otimização SGD.



Fonte: Autor

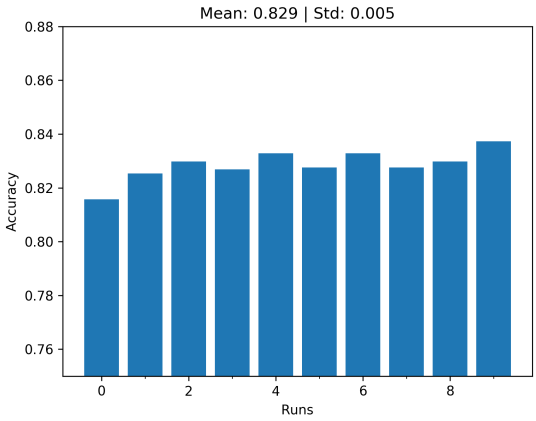
Como foi computado a partir dos resultados indicados pela Figura 24, a média de 81,8% ($\pm$ 0,3%) de acurácia através da otimização SGD, não superam a Adam que foi apresentada na Figura 23 (a) que obteve a média de 85,6% ($\pm$ 0,4%).

5.4 COMPARAÇÃO ENTRE DIFERENTES QUANTIDADES DE ELEMENTOS PROCESSADORES NA CAMADA ESCONDIDA DA MLP

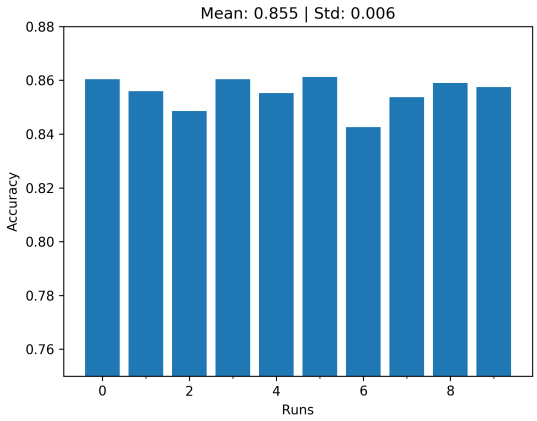
Agora com a função de ativação e técnica de otimização definidas, foram investigadas diferentes quantidades de elementos processadores na camada escondida da MLP, como ilustra a Figura 25.

Foi identificado com base nos resultados da Figura 25, com 100 elementos processadores obteve-se uma média de 82,9% ($\pm$ 0,5%), com 500 a média ficou em 85,5% ($\pm$ 0,5%) com 1000, Figura 23 (a), alcançou 85,6% ($\pm$ 0,4%) e com 2000 obteve 85,7% ($\pm$ 0,4%). Neste caso, foi definido 1000 elementos processadores para a camada escondida, pois 2000 alcançou o mesmo resultado, porém utilizando mais recursos computacionais; além disso, há o fato de que a partir de 500 elementos, a acurácia convergiu para a mesma região de valores.

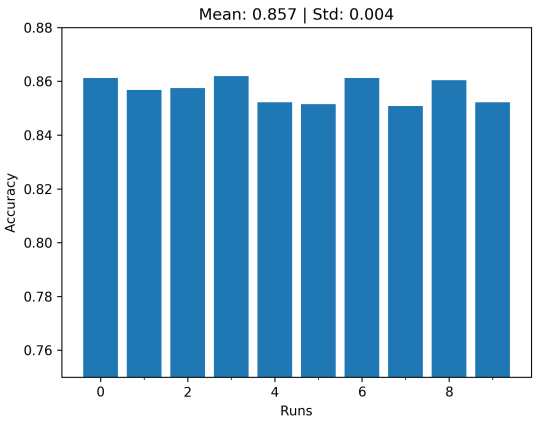
Figura 25 – Experimento para comparar os resultados na base MIT Indoor 67 com quantidades de elementos processadores na camada escondida da MLP: 100 (a), 500 (b) e 2000 (c).



(a) 100 elementos processadores



(b) 500 elementos processadores



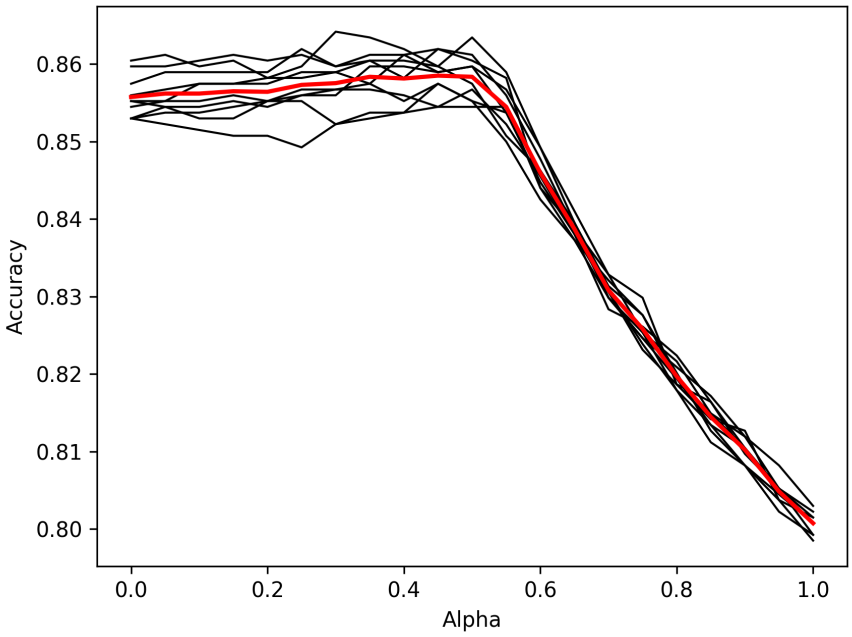
(c) 2000 elementos processadores

Fonte: Autor

5.5 AJUSTE DO PARÂMETRO α (ALPHA)

Neste experimento, foi colocado em teste a eficiência de cada *codelet* de atenção na base MIT Indoor 67, especificados na Seção 4.7, considerando as melhores parametrizações calculadas nos experimentos anteriores, isto é ilustrado na Figura 26.

Figura 26 – Performance do modelo em função do parâmetro α na base MIT Indoor 67 em 10 iterações de teste (a linha em vermelho indica a curva média entre as iterações), com passo de 0,05.



Fonte: Autor

Neste experimento, foi utilizado o SVM com *kernel* RBF, 1000 elementos processadores em uma única camada escondida da MLP, sendo que cada elemento processador utilizou a função de ativação sigmóide e o método Adam como otimizador. Além disso, foram executadas 10 iterações do teste, de modo a avaliar o impacto dos fatores aleatórios do modelo.

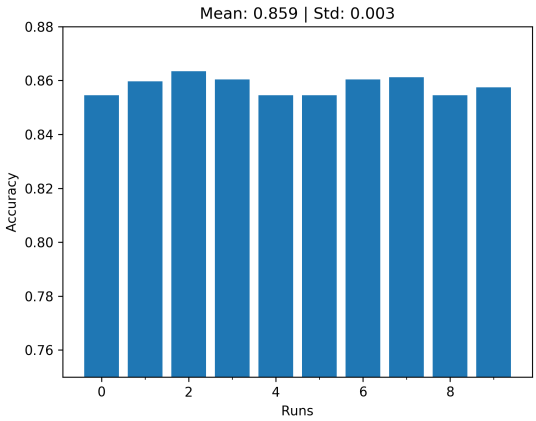
Como a Figura 26 indica, os melhores resultados (eixo y) foram obtidos com o valor de α $\sim$0,5 (eixo x), uma vez que a curva média (linha vermelha) das 10 iterações (linhas em preto) têm um pico nesta região. Com isso, é possível afirmar que os melhores resultados foram obtidos quando ambos os métodos são utilizados em conjunto; ao invés de quando cada método é utilizado sozinho, isto é $\alpha = 0$ (somente o MLP) e $\alpha = 1$ (somente a SVM).

5.6 AJUSTE DE THRESHOLD DA YOLO

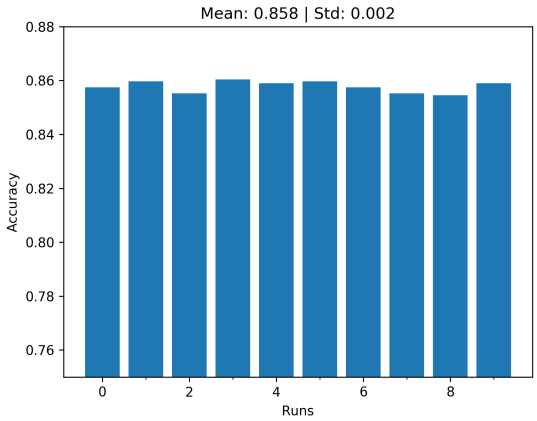
Como os parâmetros dos *codelets* foram ajustados, o parâmetro restante a ser definido é o *threshold* utilizado na YOLO dentro do EPAM, como explicado na Seção 4.4 e o resultado está ilustrado na Figura 27. Neste experimento foi considerado $\alpha = 0{,}5$. Este experimento

tem a motivação de avaliar a sensibilidade da YOLO para com a inferência final do modelo proposto.

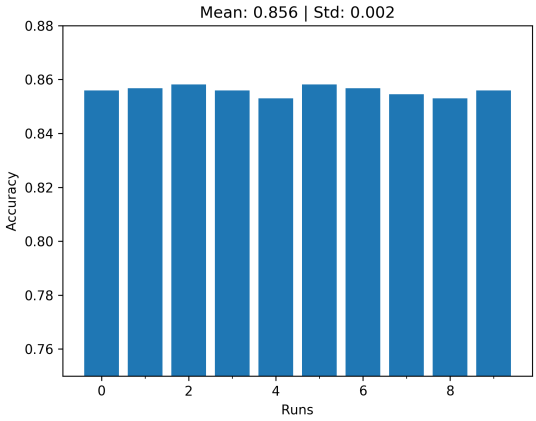
Figura 27 – Experimento para comparar os resultados na base MIT Indoor 67 com diferentes *thresholds* na YOLO utilizada no EPAM: 0,5 (a), 0,7 (b) e 0,9 (c).



(a) *Threshold* = 0,5



(b) *Threshold* = 0,7



(c) *Threshold* = 0,9

Fonte: Autor

Como pode ser concluído a partir da Figura 27, onde nenhuma das configurações se destacou, o *threshold* igual a 0,5 que alcançou a média de 85,9% de acurácia ($\pm$ 0,3%), seguido do 0,7 com 85,8% ($\pm$ 0,2%) e o 0,9 com 85,6% ($\pm$ 0,2%).

5.7 CONTRIBUIÇÃO DE CADA MEMÓRIA PARA A INFERÊNCIA

Uma vez que o modelo teve seus parâmetros ajustados, foi testada a contribuição de cada memória para a classificação, como indica a Tabela 3. Além disso, os resultados do modelo proposto são comparados com os principais trabalhos relacionados, como citado anteriormente.

Tabela 3 – Sumário de resultados com diferentes configurações de memórias utilizadas para a base MIT Indoor 67: EPAM (A), memória espacial (B) e memória declarativa (C).

Configuração	**Acurácia média**
A+B+C	85,9% ($\pm$ 0,3%)
A	30,2% ($\pm$ 0,3%)
B	77,6% ($\pm$ 0,3%)
C	85,3% ($\pm$ 0,3%)
A+B	79% ($\pm$ 0,3%)
B+C	85,7% ($\pm$ 0,2%)
A+C	85,9% ($\pm$ 0,2%)
Cheng et al. (2018)	86,76%
Zhao e Larson (2018)	-

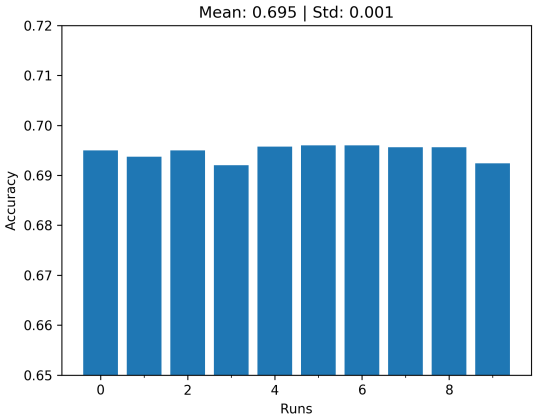
Fonte: Autor

Como pode ser visto na Tabela 3, o modelo proposto apresentou resultados comparáveis ao estado-da-arte na base MIT Indoor 67. Outra informação relevante é que cada memória tem uma contribuição diferente para a inferência (cognição do modelo proposto); além disso, com a utilização do EPAM e a memória declarativa, a memória espacial pode ser descartada sem prejudicar a performance da classificação (mesmo tendo uma acurácia maior que o EPAM). Assim fica evidente que o modelo proposto baseado em LIDA são diferentes tipos de informações em suas memórias e quando combinadas mais assertivo.

5.8 DESEMPENHO NA BASE SUN 397

Com as parametrizações identificadas na base de dados MIT Indoor 67, foram executadas as 10 configurações de treino e teste da base SUN 397, descritas na Seção 4.1 e como mostra a Figura 28.

Figura 28 – Experimento para inferir a acurácia do modelo na base SUN 397.



Fonte: Autor

No experimento referente à base de dados SUN 397 (Figura 28), o modelo proposto obteve a média de 69,5% ($\pm$ 0,1%). Este resultado é comparável ao estado-da-arte proposto por Zhao e Larson (2018) que alcançou 73,59% de acurácia, como também ao trabalho de Cheng et al. (2018) que atingiu 73,41%.

5.9 COMPARAÇÃO ENTRE CLASSES

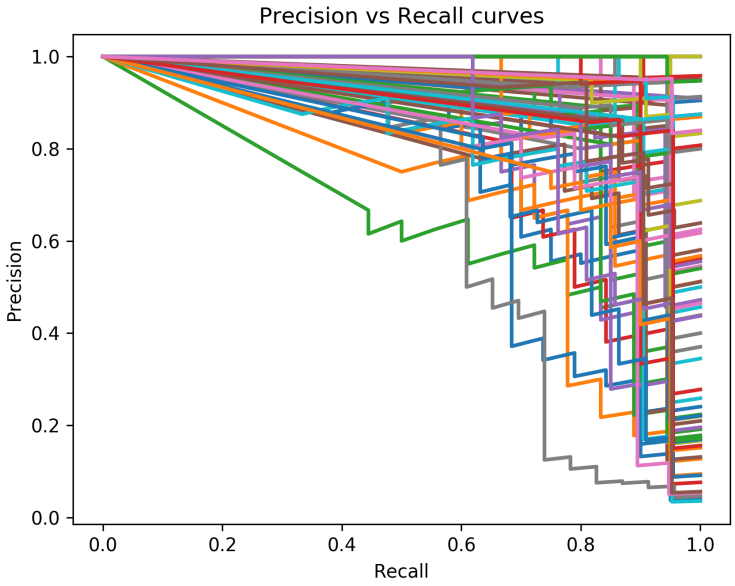
De modo a reforçar as informações somente baseadas em acurácia, foram realizados experimentos com base em *Precision-Recall*, além dos resultados de acurácia, este experimento (Figuras 29, 30, 31 e 32) apresenta os resultados de *Precision-Recall* para ambas as bases de dados utilizadas neste trabalho.

Para a base MIT Indoor 67, a Figura 29 apresenta as curvas de *Precision-Recall* para cada classe da base (diferenciadas pela cor). Além disso, a Figura 31 apresenta a curva de micro-média referente as curvas de cada classe da base, onde pode ser observado que esta curva média possui uma área de 85% do plano. Os detalhes classe-a-classe deste teste pode visto no Anexo A, onde pode ser identificado que a média de precisão ponderada pelos número de amostras por classe chega a 86,6%.

Enquanto na base de dados SUN 397, indicada pela Figura 30, as curvas estão mais balanceadas, no sentido de existirem algumas ruins, outras medianas e também as que possuem alta precisão e com grande revocação. Em adicional, a Figura 32 mostra a curva de micro-média referente às curvas de cada classe da base de dados SUN 397, onde pode ser observado que esta

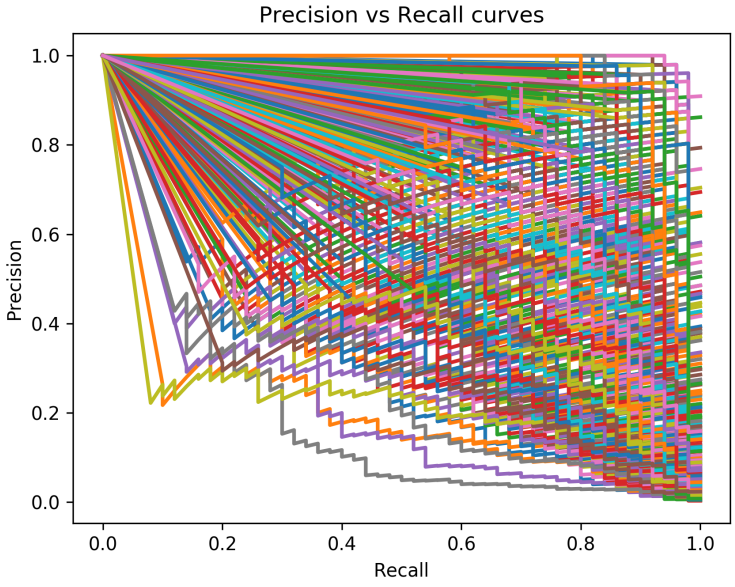
curva média possui uma área de 65% do plano. Os detalhes classe-a-classe deste teste pode visto no Anexo B.

Figura 29 – Precision-Recall para a base MIT Indoor 67.



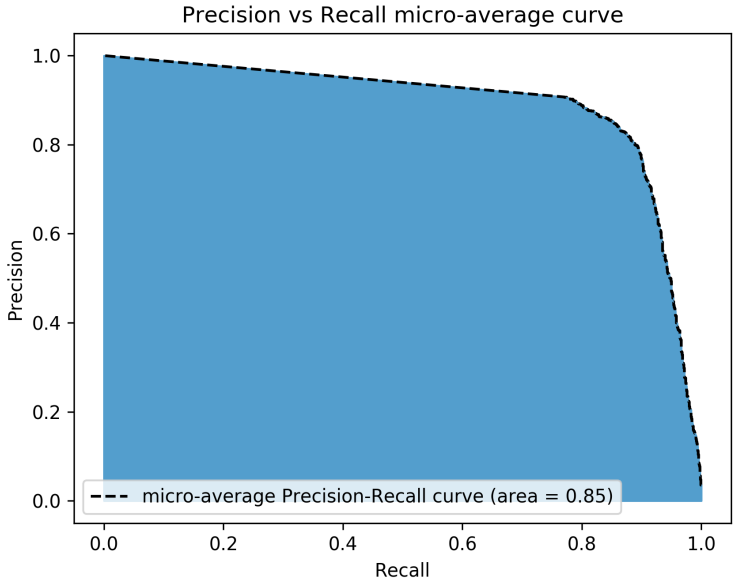
Fonte: Autor

Figura 30 – Precision-Recall para a base SUN 397.



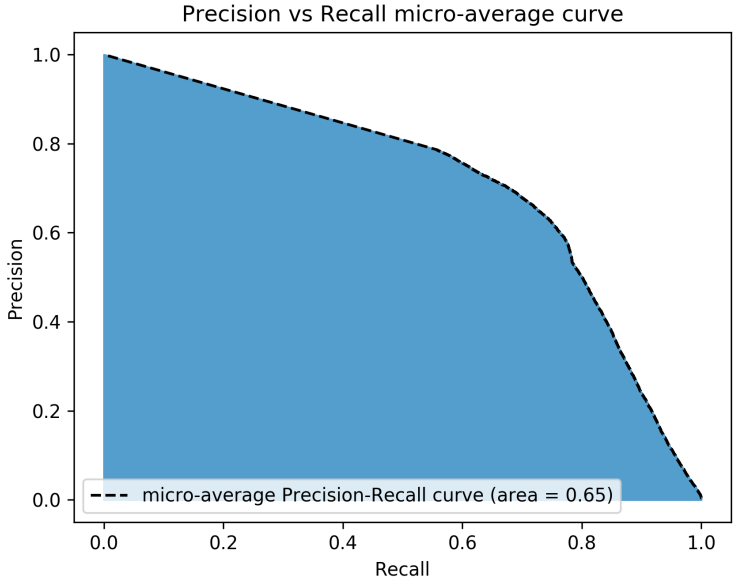
Fonte: Autor

Figura 31 – Micro-média para o Precision-Recall na base SUN 397 (desvio padrão calculado com as curvas polinomiais aproximadas de segundo grau de cada classe: 23,1%).



Fonte: Autor

Figura 32 – Micro-média para o Precision-Recall na base SUN 397 (desvio padrão calculado com as curvas polinomiais aproximadas de segundo grau de cada classe: 30,7%).



Fonte: Autor

6 DISCUSSÃO

Neste capítulo serão apontados os tópicos gerais de discussão referentes ao modelo proposto.

Uma vez que o modelo proposto visa integrar métodos de classificação de cenas, já existentes na literatura, à arquitetura LIDA, cria-se algumas divergências com o conceito original da LIDA:

a) A primeira divergência é a utilização das redes neurais e SVM, quem impedem o modelo proposto de ser totalmente assíncrono como o conceito original da LIDA.
b) Além disso, a separação entre treino e teste diverge da LIDA, que utiliza um processo de treinamento enquanto opera (infere); ou seja, um treinamento *online*.

Estas divergências acontecem porque este trabalho busca conectar todos estes conceitos, a fim de fortalecer uma nova linha de pesquisa; ao invés de buscar puramente uma arquitetura cognitiva para a classificação de cenas.

7 CONCLUSÃO

Neste trabalho foi desenvolvido um modelo para tratar o problemas de classificação de cenas através de *bag of features* baseando-se na arquitetura cognitiva LIDA. Foram empregados métodos utilizados na área de Visão Computacional para compor o modelo proposto: *Deep Learning* e SVM.

Os testes consistiram de executar múltiplas configurações, de modo a produzir resultados mais assertivos, dado os parâmetros aleatórios existentes no modelo proposto. Além disso, foram apresentadas mais de uma métrica nos resultados, a fim de mitigar as chances de conclusões enviesadas sobre os testes.

Os resultados foram obtidos através de treinos e testes em duas bases de dados amplamente utilizada no contexto do problema deste trabalho: MIT Indoor 67 e SUN 397. Durante a experimentação, a base MIT Indoor 67 foi utilizada para definir a parametrização do modelo proposto, sendo que esta parametrização foi reutilizada para os treinos e testes da SUN 397.

A partir dos resultados de experimentos apresentados no Capítulo 5, concluí-se que o modelo proposto teve sua inferência (cognição) aprimorada ao longo dos testes apresentados. Além disso, foi avaliada a contribuição de cada memória do modelo proposto, de forma que pode ser observado como as informações coletadas ao longo do ciclo cognitivo LIDA impacta a resposta do modelo para o cenário que está inserido. Dadas todas estas informações, é possível afirmar que o modelo proposto produz resultados comparáveis ao estado-da-arte, uma vez que 85,9% de acurácia na base MIT Indoor 67 e 69,5% na base de dados SUN 397.

A classificação de cenas pode ser aplicada em diversas áreas ou situações do cotidiano, um exemplo é o auxílio à navegação autônoma de veículos ou robôs, como também funcionalidades de aplicativos de fotos que podem agrupar ou pesquisar imagens semelhantes.

Como trabalhos futuros estão alguns tópicos que não foram abordados neste trabalho, dentre eles estão a tarefa de parametrizar o modelo proposto através da base SUN 397, de modo que isso possa contribuir para uma acurácia melhor; outro ponto que pode impactar nos resultados é a ordem que os parâmetros são ajustados. Além disso, outros modelos podem ser utilizados como *codelets*, como por exemplo os modelos bayesianos. Outro ponto que pode ser investigado futuramente é o vetor do CSM (*bag of features*), de modo a entender quais as *features* são mais relevantes para a classificação. Em adicional, também pode ser investigada a distribuição de probabilidades dos dos vetores produzidos pelos *codelets* de atenção (SVM e MLP); com isso pode-se identificar os índices de confiança de cada técnica, o que impacta a

agregação de probabilidades no *Global Workspace*. Por fim, também pode ser investigada as melhores e piores classes para a inferência de cada memória do modelo proposto, de modo a identificar as forças e as fraquezas nestas inferências.

REFERÊNCIAS

ADAMS, Sam et al. Mapping the landscape of human-level artificial general intelligence. **AI magazine**, v. 33, n. 1, p. 25–42, 2012.

ALIF, Mujadded Al Rabbani; AHMED, Sabbir; HASAN, Muhammad Abul. Isolated Bangla handwritten character recognition with convolutional neural network. **2017 20th International Conference of Computer and Information Technology (ICCIT)**, IEEE, p. 1–6, 2017.

ANDERSON, John R; MATESSA, Michael; LEBIERE, Christian. ACT-R: A theory of higher level cognition and its relation to visual attention. **Human-Computer Interaction**, L. Erlbaum Associates Inc., v. 12, n. 4, p. 439–462, 1997.

ANDRILUKA, Mykhaylo; ROTH, Stefan; SCHIELE, Bernt. People-tracking-by-detection and people-detection-by-tracking. **2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 1–8, 2008.

ANDRYCHOWICZ, Marcin et al. Learning to learn by gradient descent by gradient descent. **Advances in Neural Information Processing Systems**, p. 3981–3989, 2016.

ANKENBRANDT, Carol A.; BUCKLES, Bill P.; PETRY, Frederick E. Scene recognition using genetic algorithms with semantic nets. **Pattern Recognition Letters**, v. 11, p. 285–293, 1990.

BAARS, B.J. In the theatre of consciousness: global workspace theory, a rigorous scientific theory of consciousness. **Journal of Consciousness Studies**, 1997.

BAARS, Bernard. **A Cognitive Theory of Consciousness**. Cambridge University Press, Cambridge, 1988.

BAARS, Bernard J. The conscious access hypothesis: origins and recent evidence. **Trends in Cognitive Sciences**, Elsevier, v. 6, n. 1, p. 47–52, 2002.

BARINOVA, Olga; LEMPITSKY, Victor S.; KOHLI, Pushmeet. On Detection of Multiple Object Instances Using Hough Transforms. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 2233–2240, 2010.

BROSTOW, Gabriel J.; FAUQUEUR, Julien; CIPOLLA, Roberto. Semantic Object Classes in Video: A High-Definition Ground Truth Database. **Pattern Recognition Letters**, 2008.

BROSTOW, Gabriel J. et al. Segmentation and Recognition Using Structure from Motion Point Clouds. In: COMPUTER VISION – EUROPEAN CONFERENCE ON COMPUTER VISION 2008. **ECCV**. 2008. p. 44–57.

BULÒ, Samuel Rota; KONTSCHIEDER, Peter. Neural Decision Forests for Semantic Image Labelling. **2014 IEEE Conference on Computer Vision and Pattern Recognition**, p. 81–88, 2014.

CHENG, Xiaojuan et al. Scene recognition with objectness. **Pattern Recognition**, v. 74, p. 474–487, 2018.

DENG, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. **2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2009.

DIXIT, Mandar et al. Scene classification with semantic Fisher vectors. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 2974–2983, 2015.

EITEL, Andreas et al. Multimodal deep learning for robust RGB-D object recognition. **2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, p. 681–687, 2015.

ESTEVA, Andre et al. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, Nature Publishing Group, v. 542, n. 7639, p. 115, 2017.

EVERINGHAM, M. et al. **The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results**. 2007. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

_____. **The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results**. 2012. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

FEI-FEI, Li; FERGUS, Rob; PERONA, Pietro. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. **Computer vision and Image understanding**, Elsevier, v. 106, n. 1, p. 59–70, 2007.

FRANKLIN, Stan et al. A LIDA cognitive model tutorial. **Biologically Inspired Cognitive Architectures**, Elsevier, v. 16, p. 105–130, 2016.

FRANKLIN, Stan et al. LIDA: A systems-level architecture for cognition, emotion, and learning. **IEEE Transactions on Autonomous Mental Development**, IEEE, v. 6, n. 1, p. 19–41, 2014.

GARCIA-GASULLA, Dario et al. A visual embedding for the unsupervised extraction of abstract semantics. **Cognitive Systems Research**, v. 42, p. 73–81, 2017.

GEIGER, Andreas; LENZ, Philip; URTASUN, Raquel. Are we ready for autonomous driving? The KITTI vision benchmark suite. **2012 IEEE Conference on Computer Vision and Pattern Recognition**, p. 3354–3361, 2012.

GRIFFIN, Gregory; HOLUB, Alex; PERONA, Pietro. Caltech-256 object category dataset. California Institute of Technology, 2007.

HE, Kaiming et al. Deep residual learning for image recognition. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 770–778, 2016.

HE, Tong et al. Bag of Tricks for Image Classification with Convolutional Neural Networks. **arXiv preprint arXiv:1812.01187**, 2018.

HERRANZ, Luis; JIANG, Shuqiang; LI, Xiangyang. Scene Recognition with CNNs: Objects, Scales and Dataset Bias. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 571–579, 2016.

HINTON, Geoffrey E; SALAKHUTDINOV, Ruslan R. Reducing the dimensionality of data with neural networks. **Science**, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

HOMOD, Raad Z et al. Gradient auto-tuned Takagi–Sugeno Fuzzy Forward control of a HVAC system using predicted mean vote index. **Energy and Buildings**, Elsevier, v. 49, p. 254–267, 2012.

HSU, Chih-Wei; CHANG, Chih-Chung; LIN, Chih-Jen. A Practical Guide to Support Vector Classication. **National Taiwan University**, 2008.

HUANG, Gary B. et al. **Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments**. Out. 2007.

IOFFE, Sergey; SZEGEDY, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015.

JIA, Yangqing et al. Caffe: Convolutional architecture for fast feature embedding. **Proceedings of the 22nd ACM international conference on Multimedia**, p. 675–678, 2014.

KANELLAKIS, Christoforos; NIKOLAKOPOULOS, George. Survey on computer vision for uavs: Current developments and trends. **Journal of Intelligent & Robotic Systems**, Springer, v. 87, n. 1, p. 141–168, 2017.

KARPATHY, Andrej; JOHNSON, Justin. **CS231n Convolutional Neural Networks for Visual Recognition**. 2019. Disponível em: <http://cs231n.github.io/convolutional-networks/>. Acesso em: 2 jun. 2019.

KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KNERR, Stefan; PERSONNAZ, Léon; DREYFUS, Gérard. Single-layer learning revisited: a stepwise procedure for building and training a neural network. **Neurocomputing**, Springer, p. 41–50, 1990.

KONTSCHIEDER, Peter et al. Structured Labels in Random Forests for Semantic Labelling and Object Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 36, p. 2104–2116, 2014.

KORC, Filip; FÖRSTNER, Wolfgang. eTRIMS Image Database for interpreting images of man-made scenes. **Dept. of Photogrammetry, University of Bonn, Tech. Rep. TR-IGG-P-2009-01**, 2009.

KOTSERUBA, Iuliia; TSOTSOS, John K. A review of 40 years of cognitive architecture research: Core cognitive abilities and practical applications. **arXiv preprint arXiv:1610.08602**, 2018.

KRIZHEVSKY, Alex; HINTON, Geoffrey. **Learning multiple layers of features from tiny images**. 2009.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. **Commun. ACM**, v. 60, p. 84–90, 2012.

LAI, Kevin et al. A large-scale hierarchical multi-view RGB-D object dataset. **2011 IEEE International Conference on Robotics and Automation**, p. 1817–1824, 2011.

_____. Detection-based object labeling in 3D scenes. **2012 IEEE International Conference on Robotics and Automation**, p. 1330–1337, 2012.

LAZEBNIK, Svetlana; SCHMID, Cordelia; PONCE, Jean. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. **2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)**, v. 2, p. 2169–2178, 2006.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LECUN, Yann et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, p. 2278–2324, 1998.

LI, Li-Jia; FEI-FEI, Li. What, where and who? Classifying events by scene and object recognition. **2007 IEEE 11th International Conference on Computer Vision**, p. 1–8, 2007.

LIANG, Ming; HU, Xiaolin. Recurrent convolutional neural network for object recognition. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 3367–3375, 2015.

LIN, Tsung-Yi et al. Microsoft COCO: Common Objects in Context. In: SPRINGER INTERNATIONAL PUBLISHING. **Computer Vision – ECCV 2014**. Edição: David Fleet, 2014. p. 740–755.

LIU, Shaopeng; TIAN, Guohui; XU, Yuan. A novel scene classification model combining ResNet based transfer learning and data augmentation with a filter. **Neurocomputing**, Elsevier, 2019.

LIU, Song et al. Adaptive hierarchical multi-class SVM classifier for texture-based image classification. **2005 IEEE International Conference on Multimedia and Expo**, 2005.

MADL, Tamas et al. Towards real-world capable spatial memory in the LIDA cognitive architecture. **Biologically Inspired Cognitive Architectures**, 2016.

NARENDRA, Kumpati S; PARTHASARATHY, Kannan. Gradient methods for the optimization of dynamical systems containing neural networks. **IEEE Transactions on Neural Networks**, IEEE, v. 2, n. 2, p. 252–262, 1991.

NETZER, Yuval et al. Reading digits in natural images with unsupervised feature learning. **NIPS**, 2011.

NG, Gee Wah et al. Scene understanding using DSO Cognitive Architecture. **2012 15th International Conference on Information Fusion**, p. 2277–2284, 2012.

NOWOZIN, Sebastian et al. Decision tree fields. **2011 International Conference on Computer Vision**, p. 1668–1675, 2011.

O'SHEA, Keiron; NASH, Ryan. An Introduction to Convolutional Neural Networks. **ArXiv e-prints**, nov. 2015.

PATTERSON, Genevieve; HAYS, James. Sun attribute database: Discovering, annotating, and recognizing scene attributes. **2012 IEEE Conference on Computer Vision and Pattern Recognition**, p. 2751–2758, 2012.

QUATTONI, Ariadna; TORRALBA, Antonio. Recognizing indoor scenes. **2009 IEEE Conference on Computer Vision and Pattern Recognition**, p. 413–420, 2009.

REDMON, Joseph et al. You only look once: Unified, real-time object detection. **Proceedings of the IEEE conference on computer vision and pattern recognition**, p. 779–788, 2016.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUDER, Sebastian. An overview of gradient descent optimization algorithms. **CoRR**, abs/1609.04747, 2016. arXiv: 1609.04747. Disponível em: <http://arxiv.org/abs/1609.04747>.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

SANJEEVI, Madhu. **Chapter 3: Support Vector machine with Math**. 2017. Disponível em: <https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be>. Acesso em: 30 jun. 2019.

SHOTTON, Jamie et al. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation. In: COMPUTER VISION – EUROPEAN CONFERENCE ON COMPUTER VISION 2006. **ECCV**. 2006.

SICRE, Ronan et al. Automatic Discovery of Discriminative Parts as a Quadratic Assignment Problem. **2017 IEEE International Conference on Computer Vision Workshops (ICCVW)**, p. 1059–1068, 2017.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. **CoRR**, abs/1409.1556, 2015.

SRIVASTAVA, Nitish et al. Dropout: A simple way to prevent neural networks from overfitting. **The Journal of Machine Learning Research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

STITSON, MO et al. Theory of support vector machines. **University of London**, v. 117, n. 827, p. 188–191, 1996.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. Springer Science & Business Media, 2010.

TAMAAZOUSTI, Youssef; BORGNE, Hervé Le; HUDELOT, Céline. MuCaLe-Net: Multi Categorical-Level Networks to Generate More Discriminating Features. **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 5282–5291, 2017.

TIAN, Yonglin et al. Training and testing object detectors with virtual images. **IEEE/CAA Journal of Automatica Sinica**, v. 5, p. 539–546, 2018.

TORREY, Lisa; SHAVLIK, Jude. Transfer learning. **Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**, IGI Global, p. 242–264, 2010.

U.S. AIR FORCE. **MSTAR Database**. 2012. Disponível em: <https://www.sdms.afrl.af.mil/index.php?collection=mstar>. Acesso em: 5 mar. 2019.

WACHS-LOPES, Guilherme Alberto. **Reconhecimento de Objetos Utilizando Percepção Multissensorial Competitiva Baseada em Redes Complexas**. 2016. Tese (Doutorado) – Centro Universitário FEI.

WANG, Limin et al. Knowledge Guided Disambiguation for Large-Scale Scene Classification With Multi-Resolution CNNs. **IEEE Transactions on Image Processing**, v. 26, p. 2055–2068, 2017.

WITTEN, Ian H et al. **Data Mining: Practical machine learning tools and techniques**. Morgan Kaufmann, 2016.

WOHLHART, Paul; LEPETIT, Vincent. Learning descriptors for object recognition and 3D pose estimation. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 3109–3118, 2015.

XIAO, Jianxiong et al. SUN database: Large-scale scene recognition from abbey to zoo. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, p. 3485–3492, 2010.

YAO, Bangpeng et al. Human action recognition by learning bases of action attributes and parts. **2011 International Conference on Computer Vision**, p. 1331–1338, 2011.

ZHAO, Baoyong; QI, Yingjian. Image classification with ant colony based support vector machine. **Proceedings of the 30th Chinese Control Conference**, p. 3260–3263, 2011.

ZHAO, Zhengyu; LARSON, Martha. From Volcano to Toyshop: Adaptive Discriminative Region Discovery for Scene Recognition. **arXiv preprint arXiv:1807.08624**, 2018.

ZHEN, Xiantong; ZHANG, Qiujing. Spatial Ensemble Kernel Learning for Scene Classification. **2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**, p. 1303–1307, 2018.

ZHOU, Bolei et al. Learning Deep Features for Scene Recognition using Places Database. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 27. **NIPS**. 2014.

ANEXO A – PRECISION-RECALL - MIT INDOOR 67

	precision	recall	f1-score	support
airport_inside	0.777778	0.700000	0.736842	20.000000
artstudio	0.777778	0.700000	0.736842	20.000000
auditorium	0.647059	0.611111	0.628571	18.000000
bakery	0.764706	0.684211	0.722222	19.000000
bar	0.736842	0.777778	0.756757	18.000000
bathroom	0.708333	0.944444	0.809524	18.000000
bedroom	0.894737	0.809524	0.850000	21.000000
bookstore	0.772727	0.850000	0.809524	20.000000
bowling	1.000000	1.000000	1.000000	20.000000
buffet	0.857143	0.900000	0.878049	20.000000
casino	0.947368	0.947368	0.947368	19.000000
children_room	0.769231	0.555556	0.645161	18.000000
church_inside	0.894737	0.894737	0.894737	19.000000
classroom	0.761905	0.888889	0.820513	18.000000
cloister	0.909091	1.000000	0.952381	20.000000
closet	0.944444	0.944444	0.944444	18.000000
clothingstore	1.000000	0.888889	0.941176	18.000000
computerroom	0.941176	0.888889	0.914286	18.000000
concert_hall	0.818182	0.900000	0.857143	20.000000
corridor	0.944444	0.809524	0.871795	21.000000
deli	0.900000	0.473684	0.620690	19.000000
dentaloffice	0.882353	0.714286	0.789474	21.000000
dining_room	0.789474	0.833333	0.810811	18.000000
elevator	0.913043	1.000000	0.954545	21.000000
fastfood_restaurant	0.833333	0.882353	0.857143	17.000000
florist	0.950000	1.000000	0.974359	19.000000
gameroom	1.000000	0.950000	0.974359	20.000000
garage	0.894737	0.944444	0.918919	18.000000
greenhouse	1.000000	0.950000	0.974359	20.000000

	precision	recall	f1-score	support
grocerystore	0.826087	0.904762	0.863636	21.000000
gym	1.000000	0.944444	0.971429	18.000000
hairsalon	0.850000	0.809524	0.829268	21.000000
hospitalroom	0.782609	0.900000	0.837209	20.000000
inside_bus	0.958333	1.000000	0.978723	23.000000
inside_subway	1.000000	0.952381	0.975610	21.000000
jewelleryshop	0.703704	0.863636	0.775510	22.000000
kindergarden	0.863636	0.950000	0.904762	20.000000
kitchen	0.913043	1.000000	0.954545	21.000000
laboratorywet	0.916667	1.000000	0.956522	22.000000
laundromat	0.952381	0.909091	0.930233	22.000000
library	0.842105	0.800000	0.820513	20.000000
livingroom	0.850000	0.850000	0.850000	20.000000
lobby	0.850000	0.850000	0.850000	20.000000
locker_room	1.000000	0.904762	0.950000	21.000000
mall	0.842105	0.800000	0.820513	20.000000
meeting_room	0.750000	0.954545	0.840000	22.000000
movietheater	0.833333	0.750000	0.789474	20.000000
museum	0.684211	0.565217	0.619048	23.000000
nursery	1.000000	0.900000	0.947368	20.000000
office	0.833333	0.714286	0.769231	21.000000
operating_room	0.789474	0.789474	0.789474	19.000000
pantry	0.863636	0.950000	0.904762	20.000000
poolinside	1.000000	1.000000	1.000000	20.000000
prisoncell	0.947368	0.900000	0.923077	20.000000
restaurant	0.800000	0.800000	0.800000	20.000000
restaurant_kitchen	0.807692	0.913043	0.857143	23.000000
shoeshop	0.714286	0.789474	0.750000	19.000000
stairscase	0.904762	0.950000	0.926829	20.000000

	precision	recall	f1-score	support
studiomusic	0.950000	1.000000	0.974359	19.000000
subway	0.875000	1.000000	0.933333	21.000000
toystore	0.681818	0.681818	0.681818	22.000000
trainstation	0.761905	0.800000	0.780488	20.000000
tv_studio	1.000000	1.000000	1.000000	18.000000
videostore	0.863636	0.863636	0.863636	22.000000
waitingroom	0.933333	0.666667	0.777778	21.000000
warehouse	0.950000	0.904762	0.926829	21.000000
winecellar	0.875000	1.000000	0.933333	21.000000
accuracy	0.862687	0.862687	0.862687	0.862687
macro avg	0.865673	0.862313	0.860425	1340.000000
weighted avg	0.865658	0.862687	0.860627	1340.000000

ANEXO B – PRECISION-RECALL - SUN 397

	precision	recall	f1-score	support
a@abbey	0.533333	0.320000	0.400000	50.000000
a@airplane_cabin	0.913043	0.840000	0.875000	50.000000
a@airport_terminal	0.460000	0.460000	0.460000	50.000000
a@alley	0.720930	0.620000	0.666667	50.000000
a@amphitheater	0.895833	0.860000	0.877551	50.000000
a@amusement_arcade	0.655172	0.760000	0.703704	50.000000
a@amusement_park	0.777778	0.700000	0.736842	50.000000
a@anechoic_chamber	0.811321	0.860000	0.834951	50.000000
a@apartment_building@outdoor	0.380000	0.380000	0.380000	50.000000
a@apse@indoor	0.478873	0.680000	0.561983	50.000000
a@aquarium	0.857143	0.960000	0.905660	50.000000
a@aqueduct	0.795918	0.780000	0.787879	50.000000
a@arch	0.763158	0.580000	0.659091	50.000000
a@archive	0.769231	0.600000	0.674157	50.000000
a@arrival_gate@outdoor	0.866667	0.780000	0.821053	50.000000
a@art_gallery	0.654545	0.720000	0.685714	50.000000
a@art_school	0.500000	0.560000	0.528302	50.000000
a@art_studio	0.454545	0.400000	0.425532	50.000000
a@assembly_line	0.522727	0.460000	0.489362	50.000000
a@athletic_field@outdoor	0.547170	0.580000	0.563107	50.000000
a@atrium@public	0.717949	0.560000	0.629213	50.000000
a@attic	0.765957	0.720000	0.742268	50.000000
a@auditorium	0.604651	0.520000	0.559140	50.000000
a@auto_factory	0.745455	0.820000	0.780952	50.000000
b@badlands	0.712121	0.940000	0.810345	50.000000
b@badminton_court@indoor	0.769231	0.600000	0.674157	50.000000
b@baggage_claim	0.695652	0.640000	0.666667	50.000000
b@bakery@shop	0.557377	0.680000	0.612613	50.000000
b@balcony@exterior	0.734694	0.720000	0.727273	50.000000
b@balcony@interior	0.765957	0.720000	0.742268	50.000000

	precision	recall	f1-score	support
b@ball_pit	0.979592	0.960000	0.969697	50.000000
b@ballroom	0.644444	0.580000	0.610526	50.000000
b@bamboo_forest	0.890909	0.980000	0.933333	50.000000
b@banquet_hall	0.622951	0.760000	0.684685	50.000000
b@bar	0.425926	0.460000	0.442308	50.000000
b@barn	0.700000	0.700000	0.700000	50.000000
b@barndoor	0.745455	0.820000	0.780952	50.000000
b@baseball_field	0.595745	0.560000	0.577320	50.000000
b@basement	0.725000	0.580000	0.644444	50.000000
b@basilica	0.478261	0.440000	0.458333	50.000000
b@basketball_court@outdoor	0.700000	0.560000	0.622222	50.000000
b@bathroom	0.795918	0.780000	0.787879	50.000000
b@batters_box	0.937500	0.900000	0.918367	50.000000
b@bayou	0.651163	0.560000	0.602151	50.000000
b@bazaar@indoor	0.731707	0.600000	0.659341	50.000000
b@bazaar@outdoor	0.632653	0.620000	0.626263	50.000000
b@beach	0.682927	0.560000	0.615385	50.000000
b@beauty_salon	0.808511	0.760000	0.783505	50.000000
b@bedroom	0.500000	0.480000	0.489796	50.000000
b@berth	0.701754	0.800000	0.747664	50.000000
b@biology_laboratory	0.387755	0.380000	0.383838	50.000000
b@bistro@indoor	0.292683	0.240000	0.263736	50.000000
b@boardwalk	0.884615	0.920000	0.901961	50.000000
b@boat_deck	0.666667	0.560000	0.608696	50.000000
b@boathouse	0.660000	0.660000	0.660000	50.000000
b@bookstore	0.777778	0.840000	0.807692	50.000000
b@booth@indoor	0.808511	0.760000	0.783505	50.000000
b@botanical_garden	0.500000	0.360000	0.418605	50.000000
b@bow_window@indoor	0.739130	0.680000	0.708333	50.000000
b@bow_window@outdoor	0.851852	0.920000	0.884615	50.000000

	precision	recall	f1-score	support
b@bowling_alley	0.957447	0.900000	0.927835	50.000000
b@boxing_ring	0.780488	0.640000	0.703297	50.000000
b@brewery@indoor	0.684211	0.780000	0.728972	50.000000
b@bridge	0.821429	0.460000	0.589744	50.000000
b@building_facade	0.608696	0.560000	0.583333	50.000000
b@bullring	0.959184	0.940000	0.949495	50.000000
b@burial_chamber	0.607843	0.620000	0.613861	50.000000
b@bus_interior	0.823529	0.840000	0.831683	50.000000
b@butchers_shop	0.680000	0.680000	0.680000	50.000000
b@butte	0.683333	0.820000	0.745455	50.000000
c@cabin@outdoor	0.681818	0.600000	0.638298	50.000000
c@cafeteria	0.666667	0.760000	0.710280	50.000000
c@campsite	0.673913	0.620000	0.645833	50.000000
c@campus	0.416667	0.300000	0.348837	50.000000
c@canal@natural	0.530612	0.520000	0.525253	50.000000
c@canal@urban	0.725490	0.740000	0.732673	50.000000
c@candy_store	0.627907	0.540000	0.580645	50.000000
c@canyon	0.822222	0.740000	0.778947	50.000000
c@car_interior@backseat	0.796296	0.860000	0.826923	50.000000
c@car_interior@frontseat	0.862745	0.880000	0.871287	50.000000
c@carrousel	0.938776	0.920000	0.929293	50.000000
c@casino@indoor	0.673913	0.620000	0.645833	50.000000
c@castle	0.471698	0.500000	0.485437	50.000000
c@catacomb	0.654545	0.720000	0.685714	50.000000
c@cathedral@indoor	0.527273	0.580000	0.552381	50.000000
c@cathedral@outdoor	0.431373	0.440000	0.435644	50.000000
c@cavern@indoor	0.771930	0.880000	0.822430	50.000000
c@cemetery	0.745763	0.880000	0.807339	50.000000
c@chalet	0.594595	0.440000	0.505747	50.000000
c@cheese_factory	0.711111	0.640000	0.673684	50.000000

	precision	recall	f1-score	support
c@chemistry_lab	0.578947	0.440000	0.500000	50.000000
c@chicken_coop@indoor	0.649123	0.740000	0.691589	50.000000
c@chicken_coop@outdoor	0.682927	0.560000	0.615385	50.000000
c@childs_room	0.596491	0.680000	0.635514	50.000000
c@church@indoor	0.571429	0.480000	0.521739	50.000000
c@church@outdoor	0.406250	0.260000	0.317073	50.000000
c@classroom	0.523810	0.440000	0.478261	50.000000
c@clean_room	0.580000	0.580000	0.580000	50.000000
c@cliff	0.820513	0.640000	0.719101	50.000000
c@cloister@indoor	0.692308	0.900000	0.782609	50.000000
c@closet	0.888889	0.960000	0.923077	50.000000
c@clothing_store	0.541667	0.520000	0.530612	50.000000
c@coast	0.525000	0.420000	0.466667	50.000000
c@cockpit	0.937500	0.900000	0.918367	50.000000
c@coffee_shop	0.311111	0.280000	0.294737	50.000000
c@computer_room	0.773585	0.820000	0.796117	50.000000
c@conference_center	0.750000	0.840000	0.792453	50.000000
c@conference_room	0.760000	0.760000	0.760000	50.000000
c@construction_site	0.403846	0.420000	0.411765	50.000000
c@control_room	0.672727	0.740000	0.704762	50.000000
c@control_tower@outdoor	0.837209	0.720000	0.774194	50.000000
c@corn_field	0.750000	0.600000	0.666667	50.000000
c@corral	0.731707	0.600000	0.659341	50.000000
c@corridor	0.700000	0.700000	0.700000	50.000000
c@cottage_garden	0.616667	0.740000	0.672727	50.000000
c@courthouse	0.596491	0.680000	0.635514	50.000000
c@courtroom	0.734694	0.720000	0.727273	50.000000
c@courtyard	0.625000	0.500000	0.555556	50.000000
c@covered_bridge@exterior	0.800000	0.800000	0.800000	50.000000
c@creek	0.571429	0.560000	0.565657	50.000000

	precision	recall	f1-score	support
c@crevasse	0.906977	0.780000	0.838710	50.000000
c@crosswalk	0.795918	0.780000	0.787879	50.000000
c@cubicle@office	0.863636	0.760000	0.808511	50.000000
d@dam	0.839286	0.940000	0.886792	50.000000
d@delicatessen	0.421053	0.320000	0.363636	50.000000
d@dentists_office	0.772727	0.680000	0.723404	50.000000
d@desert@sand	0.854167	0.820000	0.836735	50.000000
d@desert@vegetation	0.655172	0.760000	0.703704	50.000000
d@diner@indoor	0.685714	0.480000	0.564706	50.000000
d@diner@outdoor	0.687500	0.660000	0.673469	50.000000
d@dinette@home	0.535714	0.600000	0.566038	50.000000
d@dinette@vehicle	0.795918	0.780000	0.787879	50.000000
d@dining_car	0.800000	0.720000	0.757895	50.000000
d@dining_room	0.444444	0.480000	0.461538	50.000000
d@discotheque	0.741379	0.860000	0.796296	50.000000
d@dock	0.721311	0.880000	0.792793	50.000000
d@doorway@outdoor	0.754717	0.800000	0.776699	50.000000
d@dorm_room	0.596154	0.620000	0.607843	50.000000
d@driveway	0.677419	0.840000	0.750000	50.000000
d@driving_range@outdoor	0.720930	0.620000	0.666667	50.000000
d@drugstore	0.568966	0.660000	0.611111	50.000000
e@electrical_substation	0.833333	0.900000	0.865385	50.000000
e@elevator@door	0.750000	0.600000	0.666667	50.000000
e@elevator@interior	0.555556	0.800000	0.655738	50.000000
e@elevator_shaft	0.706897	0.820000	0.759259	50.000000
e@engine_room	0.796296	0.860000	0.826923	50.000000
e@escalator@indoor	0.822222	0.740000	0.778947	50.000000
e@excavation	0.595745	0.560000	0.577320	50.000000
f@factory@indoor	0.476190	0.400000	0.434783	50.000000
f@fairway	0.532258	0.660000	0.589286	50.000000

	precision	recall	f1-score	support
f@fastfood_restaurant	0.522727	0.460000	0.489362	50.000000
f@field@cultivated	0.533333	0.480000	0.505263	50.000000
f@field@wild	0.440678	0.520000	0.477064	50.000000
f@fire_escape	0.754717	0.800000	0.776699	50.000000
f@fire_station	0.759259	0.820000	0.788462	50.000000
f@firing_range@indoor	0.630435	0.580000	0.604167	50.000000
f@fishpond	0.735849	0.780000	0.757282	50.000000
f@florist_shop@indoor	0.775862	0.900000	0.833333	50.000000
f@food_court	0.545455	0.600000	0.571429	50.000000
f@forest@broadleaf	0.500000	0.460000	0.479167	50.000000
f@forest@needleleaf	0.508197	0.620000	0.558559	50.000000
f@forest_path	0.659574	0.620000	0.639175	50.000000
f@forest_road	0.758621	0.880000	0.814815	50.000000
f@formal_garden	0.700000	0.700000	0.700000	50.000000
f@fountain	0.661290	0.820000	0.732143	50.000000
g@galley	0.744681	0.700000	0.721649	50.000000
g@game_room	0.454545	0.300000	0.361446	50.000000
g@garage@indoor	0.800000	0.720000	0.757895	50.000000
g@garbage_dump	0.710526	0.540000	0.613636	50.000000
g@gas_station	0.829787	0.780000	0.804124	50.000000
g@gazebo@exterior	0.851064	0.800000	0.824742	50.000000
g@general_store@indoor	0.567568	0.420000	0.482759	50.000000
g@general_store@outdoor	0.735849	0.780000	0.757282	50.000000
g@gift_shop	0.439024	0.360000	0.395604	50.000000
g@golf_course	0.694444	0.500000	0.581395	50.000000
g@greenhouse@indoor	0.811321	0.860000	0.834951	50.000000
g@greenhouse@outdoor	0.900000	0.900000	0.900000	50.000000
g@gymnasium@indoor	0.916667	0.880000	0.897959	50.000000
h@hangar@indoor	0.754717	0.800000	0.776699	50.000000
h@hangar@outdoor	0.740741	0.800000	0.769231	50.000000

	precision	recall	f1-score	support
h@harbor	0.745455	0.820000	0.780952	50.000000
h@hayfield	0.860000	0.860000	0.860000	50.000000
h@heliport	0.904762	0.760000	0.826087	50.000000
h@herb_garden	0.641509	0.680000	0.660194	50.000000
h@highway	0.745455	0.820000	0.780952	50.000000
h@hill	0.520000	0.520000	0.520000	50.000000
h@home_office	0.531250	0.680000	0.596491	50.000000
h@hospital	0.468085	0.440000	0.453608	50.000000
h@hospital_room	0.760870	0.700000	0.729167	50.000000
h@hot_spring	0.795455	0.700000	0.744681	50.000000
h@hot_tub@outdoor	0.820513	0.640000	0.719101	50.000000
h@hotel@outdoor	0.437500	0.280000	0.341463	50.000000
h@hotel_room	0.653061	0.640000	0.646465	50.000000
h@house	0.408451	0.580000	0.479339	50.000000
h@hunting_lodge@outdoor	0.500000	0.460000	0.479167	50.000000
i@ice_cream_parlor	0.631579	0.480000	0.545455	50.000000
i@ice_floe	0.732143	0.820000	0.773585	50.000000
i@ice_shelf	0.886364	0.780000	0.829787	50.000000
i@ice_skating_rink@indoor	0.846154	0.880000	0.862745	50.000000
i@ice_skating_rink@outdoor	0.846154	0.880000	0.862745	50.000000
i@iceberg	0.862745	0.880000	0.871287	50.000000
i@igloo	0.840000	0.840000	0.840000	50.000000
i@industrial_area	0.491525	0.580000	0.532110	50.000000
i@inn@outdoor	0.400000	0.320000	0.355556	50.000000
i@islet	0.820513	0.640000	0.719101	50.000000
j@jacuzzi@indoor	0.844444	0.760000	0.800000	50.000000
j@jail@indoor	0.745098	0.760000	0.752475	50.000000
j@jail_cell	0.717391	0.660000	0.687500	50.000000
j@jewelry_shop	0.590164	0.720000	0.648649	50.000000
k@kasbah	0.816327	0.800000	0.808081	50.000000

	precision	recall	f1-score	support
k@kennel@indoor	0.869565	0.800000	0.833333	50.000000
k@kennel@outdoor	0.618182	0.680000	0.647619	50.000000
k@kindergarden_classroom	0.705882	0.720000	0.712871	50.000000
k@kitchen	0.622642	0.660000	0.640777	50.000000
k@kitchenette	0.641509	0.680000	0.660194	50.000000
l@labyrinth@outdoor	0.795455	0.700000	0.744681	50.000000
l@lake@natural	0.489796	0.480000	0.484848	50.000000
l@landfill	0.680851	0.640000	0.659794	50.000000
l@landing_deck	0.886792	0.940000	0.912621	50.000000
l@laundromat	0.977778	0.880000	0.926316	50.000000
l@lecture_room	0.500000	0.480000	0.489796	50.000000
l@library@indoor	0.739130	0.680000	0.708333	50.000000
l@library@outdoor	0.458333	0.440000	0.448980	50.000000
l@lido_deck@outdoor	0.644068	0.760000	0.697248	50.000000
l@lift_bridge	0.690909	0.760000	0.723810	50.000000
l@lighthouse	0.893617	0.840000	0.865979	50.000000
l@limousine_interior	0.867925	0.920000	0.893204	50.000000
l@living_room	0.517241	0.600000	0.555556	50.000000
l@lobby	0.588235	0.600000	0.594059	50.000000
l@lock_chamber	0.711538	0.740000	0.725490	50.000000
l@locker_room	0.785714	0.880000	0.830189	50.000000
m@mansion	0.465517	0.540000	0.500000	50.000000
m@manufactured_home	0.729167	0.700000	0.714286	50.000000
m@market@indoor	0.507692	0.660000	0.573913	50.000000
m@market@outdoor	0.723404	0.680000	0.701031	50.000000
m@marsh	0.666667	0.720000	0.692308	50.000000
m@martial_arts_gym	0.783333	0.940000	0.854545	50.000000
m@mausoleum	0.740741	0.800000	0.769231	50.000000
m@medina	0.771930	0.880000	0.822430	50.000000
m@moat@water	0.719298	0.820000	0.766355	50.000000

	precision	recall	f1-score	support
m@monastery@outdoor	0.428571	0.360000	0.391304	50.000000
m@mosque@indoor	0.666667	0.480000	0.558140	50.000000
m@mosque@outdoor	0.790698	0.680000	0.731183	50.000000
m@motel	0.607843	0.620000	0.613861	50.000000
m@mountain	0.444444	0.240000	0.311688	50.000000
m@mountain_snowy	0.610169	0.720000	0.660550	50.000000
m@movie_theater@indoor	0.878049	0.720000	0.791209	50.000000
m@museum@indoor	0.346154	0.180000	0.236842	50.000000
m@music_store	0.931818	0.820000	0.872340	50.000000
m@music_studio	0.854167	0.820000	0.836735	50.000000
n@nuclear_power_plant@outdoor	0.415385	0.540000	0.469565	50.000000
n@nursery	0.869565	0.800000	0.833333	50.000000
o@oast_house	0.807018	0.920000	0.859813	50.000000
o@observatory@outdoor	0.714286	0.700000	0.707071	50.000000
o@ocean	0.730769	0.760000	0.745098	50.000000
o@office	0.500000	0.500000	0.500000	50.000000
o@office_building	0.441176	0.300000	0.357143	50.000000
o@oil_refinery@outdoor	0.785714	0.660000	0.717391	50.000000
o@oilrig	0.937500	0.900000	0.918367	50.000000
o@operating_room	0.744681	0.700000	0.721649	50.000000
o@orchard	0.800000	0.800000	0.800000	50.000000
o@outhouse@outdoor	0.833333	0.700000	0.760870	50.000000
p@pagoda	0.897959	0.880000	0.888889	50.000000
p@palace	0.617647	0.420000	0.500000	50.000000
p@pantry	0.830189	0.880000	0.854369	50.000000
p@park	0.571429	0.480000	0.521739	50.000000
p@parking_garage@indoor	0.859649	0.980000	0.915888	50.000000
p@parking_garage@outdoor	0.703704	0.760000	0.730769	50.000000
p@parking_lot	0.596154	0.620000	0.607843	50.000000
p@parlor	0.610169	0.720000	0.660550	50.000000

	precision	recall	f1-score	support
p@pasture	0.440678	0.520000	0.477064	50.000000
p@patio	0.625000	0.700000	0.660377	50.000000
p@pavilion	0.741379	0.860000	0.796296	50.000000
p@pharmacy	0.607843	0.620000	0.613861	50.000000
p@phone_booth	0.928571	0.780000	0.847826	50.000000
p@physics_laboratory	0.442308	0.460000	0.450980	50.000000
p@picnic_area	0.543860	0.620000	0.579439	50.000000
p@pilothouse@indoor	0.833333	0.800000	0.816327	50.000000
p@planetarium@outdoor	0.666667	0.680000	0.673267	50.000000
p@playground	0.857143	0.840000	0.848485	50.000000
p@playroom	0.685185	0.740000	0.711538	50.000000
p@plaza	0.647059	0.660000	0.653465	50.000000
p@podium@indoor	0.616667	0.740000	0.672727	50.000000
p@podium@outdoor	0.976190	0.820000	0.891304	50.000000
p@pond	0.428571	0.420000	0.424242	50.000000
p@poolroom@establishment	0.685714	0.480000	0.564706	50.000000
p@poolroom@home	0.531250	0.680000	0.596491	50.000000
p@power_plant@outdoor	0.444444	0.240000	0.311688	50.000000
p@promenade_deck	0.909091	0.800000	0.851064	50.000000
p@pub@indoor	0.462963	0.500000	0.480769	50.000000
p@pulpit	0.701754	0.800000	0.747664	50.000000
p@putting_green	0.692308	0.720000	0.705882	50.000000
r@racecourse	0.952381	0.800000	0.869565	50.000000
r@raceway	0.936170	0.880000	0.907216	50.000000
r@raft	0.960000	0.960000	0.960000	50.000000
r@railroad_track	0.851064	0.800000	0.824742	50.000000
r@rainforest	0.608696	0.560000	0.583333	50.000000
r@reception	0.553571	0.620000	0.584906	50.000000
r@recreation_room	0.300000	0.240000	0.266667	50.000000
r@residential_neighborhood	0.611111	0.660000	0.634615	50.000000

	precision	recall	f1-score	support
r@restaurant	0.400000	0.360000	0.378947	50.000000
r@restaurant_kitchen	0.709091	0.780000	0.742857	50.000000
r@restaurant_patio	0.584906	0.620000	0.601942	50.000000
r@rice_paddy	0.764706	0.780000	0.772277	50.000000
r@riding_arena	0.875000	0.980000	0.924528	50.000000
r@river	0.313725	0.320000	0.316832	50.000000
r@rock_arch	0.923077	0.960000	0.941176	50.000000
r@rope_bridge	0.870370	0.940000	0.903846	50.000000
r@ruin	0.705882	0.720000	0.712871	50.000000
r@runway	0.821429	0.920000	0.867925	50.000000
s@sandbar	0.629630	0.680000	0.653846	50.000000
s@sandbox	0.880000	0.880000	0.880000	50.000000
s@sauna	0.872727	0.960000	0.914286	50.000000
s@schoolhouse	0.605263	0.460000	0.522727	50.000000
s@sea_cliff	0.586667	0.880000	0.704000	50.000000
s@server_room	0.764706	0.780000	0.772277	50.000000
s@shed	0.745098	0.760000	0.752475	50.000000
s@shoe_shop	0.750000	0.720000	0.734694	50.000000
s@shopfront	0.666667	0.800000	0.727273	50.000000
s@shopping_mall@indoor	0.722222	0.780000	0.750000	50.000000
s@shower	0.836364	0.920000	0.876190	50.000000
s@skatepark	0.862745	0.880000	0.871287	50.000000
s@ski_lodge	0.444444	0.640000	0.524590	50.000000
s@ski_resort	0.600000	0.660000	0.628571	50.000000
s@ski_slope	0.660714	0.740000	0.698113	50.000000
s@sky	0.824561	0.940000	0.878505	50.000000
s@skyscraper	0.653061	0.640000	0.646465	50.000000
s@slum	0.733333	0.880000	0.800000	50.000000
s@snowfield	0.600000	0.660000	0.628571	50.000000
s@squash_court	0.777778	0.840000	0.807692	50.000000

	precision	recall	f1-score	support
s@stable	0.780000	0.780000	0.780000	50.000000
s@stadium@baseball	0.870370	0.940000	0.903846	50.000000
s@stadium@football	0.913043	0.840000	0.875000	50.000000
s@stage@indoor	0.516667	0.620000	0.563636	50.000000
s@staircase	0.750000	0.840000	0.792453	50.000000
s@street	0.705882	0.720000	0.712871	50.000000
s@subway_interior	0.824561	0.940000	0.878505	50.000000
s@subway_station@platform	0.661290	0.820000	0.732143	50.000000
s@supermarket	0.571429	0.640000	0.603774	50.000000
s@sushi_bar	0.610169	0.720000	0.660550	50.000000
s@swamp	0.651515	0.860000	0.741379	50.000000
s@swimming_pool@indoor	0.746032	0.940000	0.831858	50.000000
s@swimming_pool@outdoor	0.785714	0.880000	0.830189	50.000000
s@synagogue@indoor	0.465116	0.400000	0.430108	50.000000
s@synagogue@outdoor	0.491803	0.600000	0.540541	50.000000
t@television_studio	0.781818	0.860000	0.819048	50.000000
t@temple@east_asia	0.735849	0.780000	0.757282	50.000000
t@temple@south_asia	0.803922	0.820000	0.811881	50.000000
t@tennis_court@indoor	0.836364	0.920000	0.876190	50.000000
t@tennis_court@outdoor	0.706897	0.820000	0.759259	50.000000
t@tent@outdoor	0.822222	0.740000	0.778947	50.000000
t@theater@indoor_procenium	0.687500	0.660000	0.673469	50.000000
t@theater@indoor_seats	0.706897	0.820000	0.759259	50.000000
t@thriftshop	0.717949	0.560000	0.629213	50.000000
t@throne_room	0.775510	0.760000	0.767677	50.000000
t@ticket_booth	0.659574	0.620000	0.639175	50.000000
t@toll_plaza	0.734694	0.720000	0.727273	50.000000
t@topiary_garden	0.886364	0.780000	0.829787	50.000000
t@tower	0.517241	0.600000	0.555556	50.000000
t@toyshop	0.534483	0.620000	0.574074	50.000000

	precision	recall	f1-score	support
t@track@outdoor	0.788462	0.820000	0.803922	50.000000
t@train_railway	0.760000	0.760000	0.760000	50.000000
t@train_station@platform	0.844444	0.760000	0.800000	50.000000
t@tree_farm	0.956522	0.880000	0.916667	50.000000
t@tree_house	0.820000	0.820000	0.820000	50.000000
t@trench	0.826923	0.860000	0.843137	50.000000
u@underwater@coral_reef	0.960784	0.980000	0.970297	50.000000
u@utility_room	0.818182	0.900000	0.857143	50.000000
v@valley	0.450980	0.460000	0.455446	50.000000
v@van_interior	0.666667	0.600000	0.631579	50.000000
v@vegetable_garden	0.641026	0.500000	0.561798	50.000000
v@veranda	0.725490	0.740000	0.732673	50.000000
v@veterinarians_office	0.849057	0.900000	0.873786	50.000000
v@viaduct	0.755102	0.740000	0.747475	50.000000
v@videostore	0.735849	0.780000	0.757282	50.000000
v@village	0.730769	0.760000	0.745098	50.000000
v@vineyard	0.703704	0.760000	0.730769	50.000000
v@volcano	0.900000	0.900000	0.900000	50.000000
v@volleyball_court@indoor	0.807018	0.920000	0.859813	50.000000
v@volleyball_court@outdoor	0.829787	0.780000	0.804124	50.000000
w@waiting_room	0.684211	0.780000	0.728972	50.000000
w@warehouse@indoor	0.647059	0.660000	0.653465	50.000000
w@water_tower	0.803922	0.820000	0.811881	50.000000
w@waterfall@block	0.583333	0.560000	0.571429	50.000000
w@waterfall@fan	0.622642	0.660000	0.640777	50.000000
w@waterfall@plunge	0.576923	0.600000	0.588235	50.000000
w@watering_hole	0.800000	0.720000	0.757895	50.000000
w@wave	0.938776	0.920000	0.929293	50.000000
w@wet_bar	0.622642	0.660000	0.640777	50.000000
w@wheat_field	0.641509	0.680000	0.660194	50.000000

	precision	recall	f1-score	support
w@wind_farm	0.905660	0.960000	0.932039	50.000000
w@windmill	0.937500	0.900000	0.918367	50.000000
w@wine_cellar@barrel_storage	0.901961	0.920000	0.910891	50.000000
w@wine_cellar@bottle_storage	0.775510	0.760000	0.767677	50.000000
w@wrestling_ring@indoor	0.700000	0.840000	0.763636	50.000000
y@yard	0.595745	0.560000	0.577320	50.000000
y@youth_hostel	0.725490	0.740000	0.732673	50.000000
accuracy	0.694962	0.694962	0.694962	0.694962
macro avg	0.695319	0.694962	0.691677	19850.000000
weighted avg	0.695319	0.694962	0.691677	19850.000000