SegSemPuzzle: Solving Jigsaw Puzzles with Semantic Segmentation

Miguel Silva Taciano, Victor Pugliese, Fabio Augusto Faria
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo (UNIFESP)
São José dos Campos, SP
Email:{miguel.taciano, pugliese, ffaria}@unifesp.br

***Abstract*—The traditional Jigsaw Puzzle is a challenging task performed by humans, mainly due to its hardness and proven to be a NP-Complete problem. Even so, recent efforts show better performance in this task using different methods involving complex computer vision and machine learning techniques. In this sense, this paper proposes new approaches based on the semantic segmentation (SS) task to solve jigsaw puzzles (visual puzzles) in reduced training scenario. To the best of our knowledge, this is the first work in the literature that uses SS for the target application. In the performed experiments, it was possible to demonstrate that *SegSemPuzzle* and *SegSemPuzzle-G* obtained excellent results when compared with other approaches existing in literature for 3×3 puzzle solving tasks.**

***Index Terms*—semantic segmentation, shortest path algorithm, deep learning, greedy algorithm, Jigsaw Puzzle;**

I. Introdução

Quebra-cabeças representam jogos que desafiam os jogadores a combinar fragmentos de imagem para formar uma imagem completa. Esses jogos apresentam uma série de desafios intrínsecos à visão computacional, como a descrição precisa de formas, correspondência parcial de contornos, identificação de padrões, extração de características e o desenvolvimento de heurísticas para a sua resolução [1]. A complexidade desses desafios, demanda uma análise minuciosa de todas as combinações possíveis a fim de alcançar a solução ótima [2].

Para ilustrar o desafio enfrentado, um quebra-cabeça de 9 peças possui 362.880 combinações em potenciais. À medida que o tamanho do jogo aumenta, a variedade de arranjos possíveis para essas peças cresce de forma fatorial, o que classifica esse problema como NP-Completo. Portanto, essa não é uma tarefa trivial para as máquinas resolverem, tornando a criação de novos algoritmos um verdadeiro desafio quando se deseja maior desempenho, menor tempo de execução e uso de memória [3].

O campo de resolução de quebra-cabeças abrange uma gama variada de aplicações e domínios. Ele pode ser aplicado para uso recreativo para ensinar fundamentos de resolução de problemas a estudantes ou em contextos mais complexos, tais como análise de dados de varreduras cerebrais, auxiliando profissionais da área médica; detecção de vírus e *malware*; e reconstrução de artefatos arqueológicos [4]–[6].

Na literatura, muitos trabalhos têm sido propostos para resolução de quebra-cabeças adotando diferentes abordagens e técnicas para essa tarefa. Doersch et al. [7] revolucionaram o campo, propondo uma arquitetura de aprendizado não-supervisionado capaz de receber duas imagens quadradas como entrada e informar qual a relação de posição entre elas em um tabuleiro de dimensões 3×3 (9 peças).

Noroozi e Favaro [8] estendem o mesmo paradigma não-supervisionado do trabalho realizado em [7], então propõem uma nova arquitetura mais eficiente conseguindo resolver os mesmos desafios em tempo reduzido de treinamento quando comparada com as abordagens existentes na literatura.

Andaló et al. [9], por sua vez, propuseram um importante artigo que sumariza três outros trabalhos anteriores [10]–[12] de seu grupo de pesquisas, em que se utiliza de uma abordagem de programação quadrática que faz o uso de métodos que envolvem resolução de equações envolvendo matrizes, buscando um valor ideal para equações definidas previamente, acopladas com o método *constrained gradient ascent* em imagens para solucionar o problema de resolução de quebra-cabeças da melhor maneira. Os autores mostram a eficácia de sua técnica chamada de PQSP para quebra-cabeças com grandes quantidades de peças.

Sholomon et al. [13] utilizou algoritmos genéticos, cujo funcionamento é inspirado nos mecanismos de evolução natural e recombinação genética, para a resolução de quebra-cabeças de grande porte, chegando a ter tabuleiros com aproximadamente 40.000 peças.

Paumard et al. publicou uma coletânea de três artigos com relação direta ao problema alvo. O primeiro artigo [14] explora a tarefa usando redes neurais profundas de entrada dupla, inspiradas pelo trabalho de [7]. Já no segundo artigo [15], o chamando método Deepzzle é uma melhoria do artigo anterior explorando agora uma nova estratégia de escolha de probabilidades com representação de grafos e utilização de caminhos mínimos de Dijkstra para determinar as melhores opções de preenchimento das peças no tabuleiro do jogo, resultando em melhoramento do desempenho do algoritmo. Finalmente, o método Alphazzle [16] foi proposto, visto um problema de alto custo computacional pela adoção de representação em grafos, eles agora utilizam uma abordagem de busca em árvore de Monte Carlo para reduzir o custo computacional e melhorar a acurácia por meio do paradigma de aprendizado por reforço.

Outras abordagens presentes na literatura relacionadas ao problema envolve a criação de quebra-cabeças como Park et al. [17], onde eles utilizam de métodos de aprendizado

profundo para a construção de novos quebra-cabeças a partir de imagens. Outro exemplo do aumento da complexidade da tarefa alvo está o trabalho de Jampy et al. [18] que buscou resolver quebra-cabeças de três dimensões usando métodos baseados em heurísticas, mostrando ser uma tarefa de alta complexidade, justamente pelo aumento de dimensão e consequentemente do aumento de parâmetros, bem como, do espaço de soluções e tempo de resolução.

Uma tarefa que tem obtido grande sucesso na área de visão computacional e que não tem sido adotada para o problema de resolução de quebra-cabeça é a chamada segmentação semântica (SS). Esta tarefa envolve a divisão de uma imagem em sub-regiões, em que cada pixel presente na imagem recebe um rótulo (classe), representando objetos, suas partes ou elementos semânticos em uma cena. SS tem sido utilizada em diferentes áreas de pesquisas como veículos autônomos, monitoramento de tráfego, contagem de pessoas, mapeamento e navegação de robôs, entre outras [19]–[21].

Com objetivo de verificar o potencial das poderosas arquiteturas de segmentação semântica em outro domínio de aplicação, neste trabalho nós propomos duas abordagem (*SegSemPuzzle* e *SegSemPuzzle-G*) de resolução de quebra-cabeças utilizando de uma famosa arquitetura chamada HRNet para criar as máscaras dos tabuleiros de quebra-cabeças de 9 peças (3×3), as quais serão utilizadas no cálculo de similaridade entre as peças do jogo.

II. Abordagens de resolução de quebra-cabeças

Esta seção explica em detalhes o problema alvo, o funcionamento das abordagens gulosas e as abordagens baseadas em caminho de custo mínimo em grafo, com ênfase nas abordagens propostas baseada em segmentação semântica.

A. Definição do Problema

O problema se concentra em jogos de quebra-cabeça de tabuleiro T de dimensão 3×3 composto de 9 peças quadradas 2D de tamanho uniforme, assim tem-se T formado de $\{P_1, \ldots, P_9\}$ (ver Figura 1-a). Além disso, adota-se uma ordem de preenchimento das posições do tabuleiro denotada por $\mathcal{T}(x)$, onde $x = \{1, \ldots, 9\}$ (ver Figura 1-b).

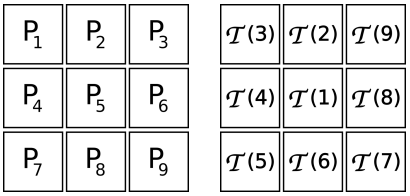


Fig. 1. Em (a) estão as peças nas suas posições corretas no tabuleiro e em (b) está o tabuleiro com a configuração sequencial de montagem do tabuleiro $\mathcal{T}(i)$, com exceção da peça central P_5 que já é conhecida.

Dado $\mathcal{T}(x)$ a próxima posição do tabuleiro da iteração x e P_i uma peça de número i, monta-se o tabuleiro seguindo a premissa que toda posição $\mathcal{T}(x)$ precisa existir uma peça P_i correspondente, tal que uma ordem de montagem do tabuleiro é respeitada, partindo de $\mathcal{T}(2)$ até $\mathcal{T}(N)$, onde N representa o número total de peças do quebra-cabeça.

A Figura 2 mostra um exemplo de etapas de resolução do problema do quebra-cabeça de dimensão 3×3, onde inicia-se com a peça central conhecida $\mathcal{T}(1) = P_5$ e o preenchimento das próximas posições $\mathcal{T}(2) = P_2$, $\mathcal{T}(3) = P_1$ até sua completa resolução $\mathcal{T}(9) = P_3$. Importante comentar que a peça P_i a ser atribuída a próxima posição $\mathcal{T}(x)$ dependerá do cálculo de similaridade da abordagem utilizada na resolução.

Pelo fato do caminho de resolução $\mathcal{T}(x)$ ser pré-determinado, a solução completa ideal para o problema do quebra-cabeça de qualquer tabuleiro T de 9 peças sempre será: $T = \{\mathcal{T}(1) = P_5, \mathcal{T}(2) = P_2, \mathcal{T}(3) = P_1, \mathcal{T}(4) = P_4, \mathcal{T}(5) = P_7, \mathcal{T}(6) = P_8, \mathcal{T}(7) = P_9, \mathcal{T}(8) = P_6, \mathcal{T}(9) = P_3\}$.

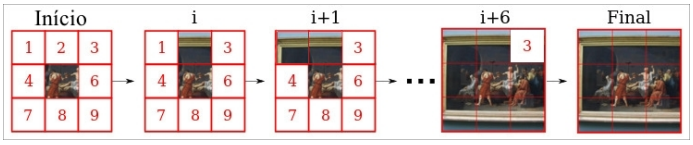


Fig. 2. Ordem de resolução do quebra-cabeça.

B. Abordagens Gulosas

As abordagens gulosas seguem o mesmo conceito de algoritmos gulosos [22], ou seja, a escolha da próxima peça P_i para ser adicionada na posição $\mathcal{T}(x)$ será aquela que tem menor custo (maior similaridade) entre todas as possibilidades de peças no momento do preenchimento (ótimo local) da posição no tabuleiro. Portanto, nesta seção serão explicadas as quatro diferentes abordagens gulosas (ResNet-50, AE, *AE4S* e SegSemPuzzle) implementadas neste trabalho, bem como, os cálculos de custo realizados de cada peça no instante de sua escolha.

O Algoritmo 1 detalha o funcionamento das abordagens gulosas de resolução de quebra-cabeças. Note que a diferença entre as abordagens gulosas está no cálculo de similaridade $Sim(P_i, \mathcal{T}(x))$ que guiará a escolha da próxima peça a ser preenchida no tabuleiro.

1) ***Rede Neural Residual (ResNet-50)****:* A arquitetura ResNet-50 é uma rede neural profunda composta por 50 camadas convolucionais e blocos/saltos residuais que têm como objetivo evitar que as informações entre camadas sejam degradadas ao longo do processo de aprendizagem [23]. Assim, esta abordagem de resolução utiliza da arquitetura ResNet-50 pré-treinada na base ImageNet [24] para extrair vetores de características de cada uma das peças P_i do problema, resultando em 9 vetores ($\mathcal{V}_{P_i}$) de tamanho 2048. O cálculo de similaridade $Sim(P_i, \mathcal{T}(x))$ é definido pela Equação 1, a qual é composta por somatório do inverso das distâncias Euclidianas $Dist(\mathcal{V}_{P_i}, \mathcal{V}_{P_z})$ entre os vetores $\mathcal{V}_{P_i}$ de cada peça P_i ainda não atribuída em alguma posição do tabuleiro e os vetores $\mathcal{V}_{P_z}$ dos vizinhos $P_z \in N_{\mathcal{T}(x)}$, onde $N_{\mathcal{T}(x)}$ é o conjunto de vizinhos já existentes no tabuleiro durante a etapa $\mathcal{T}(x)$. A quantidade de vizinhos $N_{\mathcal{T}(x)}$ depende da posição $\mathcal{T}(x)$ a ser preenchida no momento.

$$Sim(P_i, \mathcal{T}(x)) = \sum_{P_z \in N_{\mathcal{T}(x)}} \frac{1}{Dist(\mathcal{V}_{P_i}, \mathcal{V}_{P_z})} \qquad (1)$$

Algorithm 1: Abordagem Gulosa

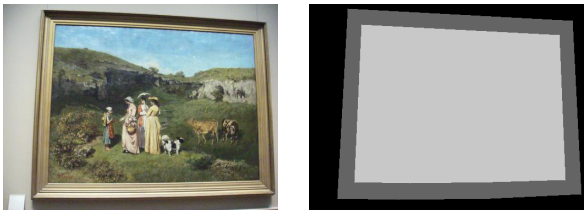
Entrada: Tabuleiro T compostos por peças $P_i, i \in \{1, \ldots, 9\}$, lista de peças remanescentes $R = \{P_1, \ldots, P_9\}$ e a lista ordenada de posições $\mathcal{T}(x), x \in \{1, \ldots, 9\}$;
Saída: Um conjunto tuplas $\mathcal{T}(x) = P_i$ que corresponde a solução;

```
T(1) = P_5;
R ← R − {P_1};
MAX ← 0;
for x ∈ [2...9] do
    for P_i ∈ R do
        Calcular Sim(P_i, T(x));
        if Sim(P_i, T(x)) ≥ MAX then
            MAX ← Sim(P_i, T(x));
            P_max ← P_i;
        end
    end
    T(x) = P_max;
end
```

*2) **Autoencoder (AE)**:* Um *autoencoder* é um tipo específico de rede neural, projetado principalmente para codificar a entrada, por meio do *encoder*, em uma representação compactada e significativa (vetor latente) e, em seguida, decodificá-la de volta, por meio de um *decoder*, para que a entrada reconstruída seja o mais semelhante possível à original [25]. Esta abordagem *AE* foi desenvolvida neste trabalho (detalhada na seção III-B) e treinada com o conjunto de treinamento da coleção. Após a etapa de treinamento da arquitetura, utiliza-se o *encoder* para extrair os vetores de características de cada peça do quebra-cabeça e o cálculo de similaridade é semelhante à Equação 1.

*3) **Autoencoder em Bordas (AE4S)**:* Nesta abordagem *AE4S*, diferente da anterior (*AE*), o cálculo da similaridade é realizado considerando 25% da região da peça em cada um dos quatro lados das peças remanescentes P_i (esquerdo: P_i^l, direito: P_i^r, cima: P_i^t e baixo: P_i^b) do jogo e os quatro lados dos seus vizinhos existentes no tabuleiro. Para o cálculo de similaridade, dadas duas peças, o valor entre elas é calculado utilizando os cantos opostos de encaixe da peça respeitando a ordem de preenchimento das posições $\mathcal{T}(x)$. Por exemplo, uma vez a posição $\mathcal{T}(1)$ preenchida, busca-se o preenchimento da posição $\mathcal{T}(2)$, por tanto o cálculo de similaridade é realizado com o lado de cima P_5^t da peça que está em $\mathcal{T}(1)$ e o lado de baixo P_i^b de todas as peças remanescentes do quebra-cabeça. Para tal, a equação 1 é similarmente utilizada nessa abordagem.

*4) **Segmentação Semântica (SegSemPuzzle)**:* Segmentação semântica é tarefa da área de visão computacional que consiste em classificar/rotular todos os pixels presentes em uma imagem de acordo com sua semântica (classe). E como resultado final do processo da tarefa está um mapa/máscara de mesmas dimensões que a imagem original com cada pixel representado por um valor inteiro da classe correspondente [26]. A Figura 3 mostra um exemplo de imagem de entrada e sua respectiva máscara ideal criada por uma abordagem de segmentação semântica.



(a) Imagem de entrada. (b) Máscara resultante.

Fig. 3. Exemplos de uma imagem de entrada e sua máscara correspondente após segmentação semântica ideal.

Dentre as arquiteturas mais bem-sucedidas da literatura, a HRNet [27] foi escolhida neste trabalho, devido a sua robustez, eficácia e capacidade de criar representações de alta resolução.

Com objetivo de deixar a arquitetura HRNet mais específica para o novo domínio de aplicação e consequentemente melhorar seus resultados, a HRNet teve que ser treinada com imagens da nova coleção. Diferentemente da configuração da HRNet no artigo original, neste trabalho, os pixels de uma imagem podem pertencer apenas a três diferentes cores/classes pré-definidas, como pode ser observado na Figura 3. Em verde, estão os pixels classificados como pintura/tela, em vermelho estão os classificados como moldura e finalmente em preto estão aqueles classificados como parede.

A Figura 4 mostra um exemplo de ajuste fino (*fine-tuning*) da arquitetura HRNet pré-treinada, onde esta aprenderá o novo domínio (obras de arte). É importante notar que essa estratégia de treinamento faz com que a arquitetura melhore a máscara resultante a medida que o processo de treinamento evolui.

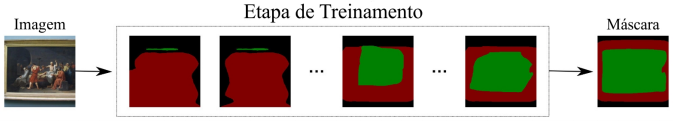


Fig. 4. A etapa treinamento da arquitetura HRNet com a imagem de entrada e a sua máscara resultante da tarefa de segmentação.

Uma vez que arquitetura HRNet está treinada para o novo domínio, ela é utilizada para segmentar cada nova imagem do conjunto de teste. Cada imagem tem sua máscara correspondente, a qual é composta de pixels com apenas 3 valores (classes). Ambas entradas (imagem e máscara) são subdivididas em 9 peças e a similaridade é dada pelo cálculo de correspondência entre os pixels das bordas (96 pixels da coluna ou linha) entre as peças remanescentes e os vizinhos da posição $\mathcal{T}(x)$ já existentes no tabuleiro.

O cálculo de correspondência utiliza a similaridade entre os lados das peças P_i (esquerdo: P_i^l, direito: P_i^r, cima: P_i^t e baixo: P_i^b). Por exemplo, fornecidas duas peças, compara-se os pixels na mesma posição relativa em lados opostos dessas peças, obtendo assim a quantidade de pixels de correspondência (mesma classe na máscara). Por fim, esse valor é dividido

pelo total de 96 pixels, obtendo a porcentagem de pixels igual correspondência entre peças.

C. Abordagens de Caminho de Custo Mínimo em Grafo

Um grafo $G = \{V, E\}$ é uma estrutura composta por dois conjuntos de elementos: os vértices V, e as arestas E. As arestas estabelecem conexões entre dois vértices u e v, as quais pode ter valores/pesos $w(u, v)$ calculados por uma função W [22]. Na teoria de grafos existe uma família de algoritmos que percorrem grafos com objetivo de encontrar o caminho $S(u, v)$ de menor custo entre dois vértices u e v de um grafo G. Um caminho $S(u, v) = \{u, \ldots, v\}$ é composto de todos os vértices que permitem que u acesse v e seu custo é calculado $W(S)$ pela soma de todos os pesos das arestas w que conectam os vértices no caminho $S(u, v)$. Importante reforçar que neste trabalho, os pesos das arestas são calculados pela similaridade entre peças utilizado por cada abordagem e definida por $Sim(P_i, \mathcal{T}(x))$. Dentre os algoritmos mais conhecidos está o de Dijkstra que inicia no vértice u e realiza uma busca do caminho de custo mínimo até vértice v dentre todos os caminhos possíveis presentes no grafo G [28].

Os Algoritmos 2 e 3, detalham o funcionamento das abordagens de resolução baseadas em caminhos mínimos de Dijkstra.

Algorithm 2: Construção do Grafo G

Entrada: Tabuleiro T compostos por peças $P_i, i \in \{1, \ldots, 9\}$, lista de peças remanescentes $R = \{P_1, \ldots, P_9\}$ e a lista ordenada de posições $\mathcal{T}(x), x \in \{1, \ldots, 9\}$;
Saída: Grafo orientado G;
$\mathcal{T}(1) = P_5$;
$R \leftarrow R - \{P_5\}$;
for *cada* $x \in [2, , \ldots, 9]$ **do**
  for *cada* $P_i \in R$ **do**
    Calcular $Sim(P_i, \mathcal{T}(x))$;
    Aresta $w(P_i, P_{\mathcal{T}(x-1)}) \leftarrow Sim(P_i, \mathcal{T}(x))$;
  end
end

A Figura 5 mostra um exemplo fictício de grafo construído, onde pode-se observar que cada nível $\mathcal{T}(x), x \in \{1 \ldots, 9\}$ representa uma etapa da reconstrução do quebra-cabeça.

Neste trabalho, duas abordagens foram desenvolvidas utilizando de representação grafo e caminho de custo mínimo (*AE4S-G* e *SegSemPuzzle-G*).

1) ***AE4S-G****:* a qual utiliza do mesmo cálculo de similaridade da abordagem gulosa *AE4S* detalhado na Seção II-B3 para preencher o peso/custo das arestas do grafo orientado totalmente construído e aplica-se o algoritmo de Dijkstra para resolução do quebra-cabeça.

2) ***SegSemPuzzle-G****:* esta por sua vez utiliza da mesma arquitetura de segmentação semântica (HRNet) e cálculo de similaridade da versão gulosa *SegSemPuzzle* explicada na Seção II-B4 para a atribuição do peso/custo das arestas do grafo direcionado e em seguida, aplica-se o algoritmo de Dijkstra para resolução do quebra-cabeça.

Algorithm 3: Resolução por Dijkstra

Entrada: Grafo orientado $G = \{V, E\}$, posição inicial do tabuleiro $\mathcal{T}(1)$, vetor auxiliar de custos W com tamanho do tabuleiro e lista de prioridade Q;
Saída: Caminho $S(P_{\mathcal{T}(1)}, P_{\mathcal{T}(9)})$;
$W[P_{\mathcal{T}(1)}] \leftarrow 0$;
$S \leftarrow \{\ \}$;
$Q \leftarrow V[G]$;
while $Q \neq \{\ \}$ **do**
  $P_u \leftarrow$ Remove-Vertice-Min-Peso(Q);
  $S \leftarrow S \cup \{P_u\}$;
  foreach *vértice* P_v *adjacente de* P_u **do**
    $aux \leftarrow W[P_u] + w(P_u, P_v)$;
    if $aux < W[P_v]$ **then**
      $W[P_v] \leftarrow aux$;
      Atualize a chave de v em Q para aux;
    end
  end
end

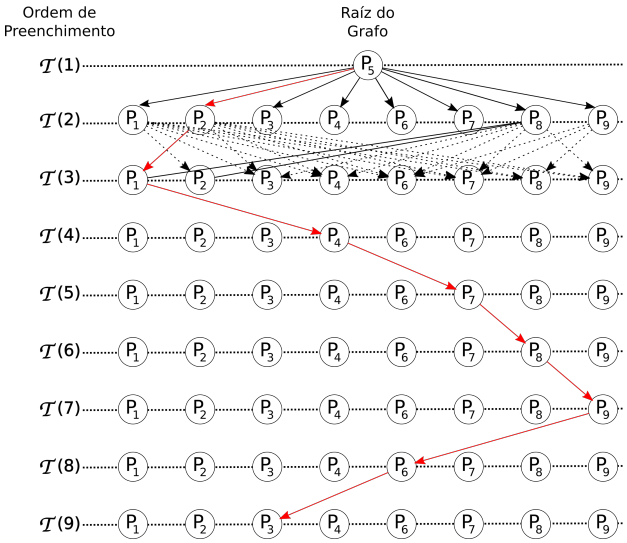


Fig. 5. O grafo orientado G construído com sua raiz na peça P_5 e todas as oito alternativas de peças do quebra-cabeça sendo escolhidas na ordem de preenchimento das posições $\mathcal{T}(x)$ colocadas à disposição no cálculo do peso de cada aresta do grafo. Em vermelho está o caminho S de menor custo ou maior similaridade que representa a solução ótima do jogo.

III. Resultados e Discussão

Nesta seção são apresentados o protocolo experimental adotado, bem como, os resultados obtidos nos experimentos realizados neste trabalho.

A. Base de Imagens

Nos experimentos realizados neste trabalho, um subconjunto de 1232 imagens de obras de artes foi selecionado da grande base de imagens MET Museum [29] composta por aproximadamente 400 mil separada em 224 mil classes imagens. Esse subconjunto de imagens foi escolhido por

ter uma semântica bem definida entre as imagens existentes com apenas três classes presentes na cena e assim, facilitar o aprendizado da arquitetura de segmentação semântica. O protocolo experimental foi pensado como um cenário restrito de amostras de treinamento, o qual teve uma escolha aleatória de apenas 100 imagens da base de treinamento original[1] para compor o conjunto de treinamento deste trabalho, 93 imagens para o conjunto de validação e para o conjunto de teste foi selecionado o conjunto inteiro de 1132 imagens da base de teste original da MET. Como as 193 imagens (treino e validação) não continham suas respectivas máscaras de segmentação, essas foram criadas de forma manual. Finalmente, seguindo o mesmo protocolo adotado no trabalho [15], as imagens sofreram uma operação de redimensionamento para resolução 288×288, resultando em 9 segmentos quadrados (peças) de 96×96.

B. Configurações Experimentais

Esta seção mostra em detalhes as configurações adotadas para cada uma das técnicas utilizadas neste trabalho.

A arquitetura ResNet-50 utilizada foi um modelo pré-treinado com pesos da ImageNet sem ajuste fino de treinamento presente na biblioteca PyTorch. Já para as arquiteturas *Autoencoder* (*AE* e *AE4S*) foi utilizada a configuração vista da Tabela I. Com essa configuração, a vetor latente é uma matriz de dimensões $4 \times 24 \times 24$ para a abordagem *AE*, resultando em compressão de 27.648 pixels para 2.304. Já para abordagem *AE4S*, o vetor latente é uma matriz de dimensões $4 \times 6 \times 24$, o que resulta em uma compressão de 6.912 pixels para 576. Para ambas abordagens, o vetor latente é equivalente a aproximadamente $8,33\%$ da quantidade total de pixels da imagem de entrada. Na abordagem de segmentação semântica foi utilizada a arquitetura HRNet com sua configuração original do artigo pré-treinada na coleção de imagens Pascal-Context [30] e aprimorada para o uso em obras de arte utilizando o conjunto de treino e validação criados neste trabalho. Finalmente, para a abordagem estado da arte (SOTA), a implementação do método Deepzzle [15] seguiu a original do artigo (rede FEN descrita em [14]) e também foi ajustado para o novo domínio.

C. Análise Comparativa entre as Abordagens

Neste trabalho, quatro abordagens gulosas (*ResNet-50*, *AE*, *AE4S* e *SegSemPuzzle*) e duas abordagens baseadas em caminhos mínimos (*AE4S* e *SegSemPuzzle-G*) foram desenvolvidas utilizando diferentes estratégias e estas foram comparadas entre si nos experimentos realizados. Além disso, o método estado da arte em resolução de quebra-cabeça, o Deepzzle [15], foi adicionado a esta análise comparativa para mostrar as reais contribuições deste trabalho.

A Tabela II mostra os resultados da análise comparativa entre todas as abordagens com relação os resultados de acurácia dos jogos e de peças para todas as 1132 imagens do conjunto de teste.

[1]Met exhibit (*train*) images & Met *query images* http://cmp.felk.cvut.cz/met/

TABLE I
CONFIGURAÇÃO DA ARQUITETURA AUTOENCODER CONSTRUÍDA PARA ESTE TRABALHO.

Encoder				
Camada	**Entrada**	**Saída**	**Kernel**	**Padding/Stride**
Conv2D_1	3	16	3×3	1
	ReLU			
	MaxPool2D, Stride 2			
Conv2D_2	16	4	3 × 3	1
	ReLU			
	MaxPool2D, Stride 2			
Decoder				
Camada	**Entrada**	**Saída**	**Kernel**	**Padding/Stride**
ConvTranspose2D_1	4	16	2 × 2	2
	ReLU			
ConvTranspose2D_2	16	3	2 × 2	2
	sigmoid			

Como pode ser observado, os resultados das abordagens têm grande variação. A abordagem *ResNet-50*, a qual utiliza do modelo da ResNet-50 e a abordagem *AE* que utiliza um *autoencoder* são em grande parte semelhantes em adotar uma descrição global da peça no cálculo de similaridade e *AE* mostra pequena vantagem. É possível notar também, grande salto de desempenho na abordagem *AE4S*, que continua utilizando de mesma arquitetura *autoencoder*, entretanto separando suas características em quatro partes (os lados da peça) e comparando os lados adjacentes no cálculo de similaridade. Além disso, as melhores abordagens neste trabalho são aquelas que utilizam da arquitetura de segmentação semântica para realizar o cálculo da similaridade entre as peças (*SegSemPuzzle* e *SegSemPuzzle-G*).

A abordagem gulosa *SegSemPuzzle* consegue resultados muito superiores as demais abordagens gulosas comparadas, sendo $76,94\%$ melhor em acurácia de jogos completos e $48,60\%$ melhor em acurácia de peças em posições corretas que a segunda melhor abordagem gulosa *AE4S*. Já a abordagem baseada em caminhos mínimos *SegSemPuzzle-G* consegue resultados surpreendentes, ganhando de todas as abordagens comparadas neste trabalho. *SegSemPuzzle-G* conseguiu ser $3,18\%$ e $1,33\%$ melhor que a abordagem *SegSemPuzzle* nas medidas de avaliação adotadas neste trabalho. Finalmente, a abordagem *SegSemPuzzle-G* consegue ser $93,73\%$ e $70,42\%$ melhor que o método Deepzzle.

As vantagens da melhor abordagem proposta neste trabalho (*SegSemPuzzle-G*) em relação ao método estado da arte da literatura, o Deepzzle [15], vão além dos resultados de eficácia. Dentre elas estão:

- **Cenário Restrito de Amostras:** devido à estratégia de ajuste fino, a abordagem *SegSemPuzzle-G* necessita de poucas amostras na etapa de treinamento. Já Deepzzle mostrou ter a necessidade de grandes bases de dados para conseguir resultados satisfatórios;
- **Diferentes Tamanhos de Tabuleiros (escalabilidade):** a abordagem *SegSemPuzzle-G* não necessita de etapa de re-treinamento da arquitetura de segmentação semântica HRNet para funcionar em tabuleiros de diferentes tamanhos em mesmo domínio semântico, permitindo maior escalabilidade. Já o Deepzzle é uma arquitetura com saída fixa e necessita ser re-treinada a cada nova configuração

TABLE II
RESULTADOS DAS ABORDAGENS DE RESOLUÇÃO DE QUEBRA-CABEÇAS. EM DESTAQUE ESTÃO AS MELHORES ABORDAGENS.

Abordagens Gulosas	**Jogos**		**Peças**	
	Completos	**Acurácia**	**Corretas**	**Acurácia**
ResNet-50	4	0,35%	2472	24,26%
AE	3	0,27%	2373	23,29%
AE4S	182	16,08%	4981	48,89%
SegSemPuzzle	**1053**	**93,02%**	**9932**	**97,49%**
Abordagens Caminhos Mínimos	**Jogos**		**Peças**	
	Completos	**Acurácia**	**Corretas**	**Acurácia**
AE4S-G	601	53,09%	7409	72,72%
SegSemPuzzle-G	**1089**	**96,20%**	**10068**	**98,82%**
Deepzzle [15]	28	2,47%	2894	28,41%

de rede;

IV. CONCLUSÃO

Neste trabalho, diferentes abordagens para resolução de quebra-cabeça foram propostas em duas famílias de algoritmos: (1) abordagens gulosas; e (2) abordagens de caminho de custo mínimo em grafo. As abordagens gulosas tentam resolver o problema do quebra-cabeça sempre utilizando das informações locais para realizar as suas escolhas das peças a serem atribuídas em uma dada posição no tabuleiro. Já as abordagens baseadas em caminho de custo mínimo em grafo, constroem o grafo de resolução do jogo e aplica-se o algoritmo de Dijkstra para encontrar a solução do quebra-cabeça. Nos experimentos realizados foi possível mostrar que as abordagens da família 2 superam suas versões gulosas em acurácias. Além disso, as abordagens que utilizam de arquitetura de segmentação semântica (*SegSemPuzzle* e *SegSemPuzzle-G*) alcançam os melhores resultados quando comparadas com as outras abordagens de mesma família. Finalmente, um método SOTA da literatura chamado Deepzzle foi comparado e não consegue resultados satisfatórios. Como trabalho futuro pode-se citar o aumento do número de imagens da coleção, mudança de domínio da aplicação para outras obras de artes diferentes de pinturas/quadros e a combinação de abordagens para melhoramento da acurácia final.

AGRADECIMENTOS

Os autores gostariam de agradecer o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa PIBIC do estudante M. Taciano e pela bolsa do V. Pugliese do Programa de Doutorado Acadêmico para Inovação (DAI) em parceria com a empresa EMBRAER. Além da Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP (#2017/25908-6 e #2018/23908-1) e Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

REFERENCES

[1] D. Kosiba et al., "An automatic jigsaw puzzle solver," in *12th ICPR*, vol. 1, 1994, pp. 616–618.

[2] R. Tybon, "Generating solutions to the jigsaw puzzle problem," *PhD thesis, Griffith University, Australia*, 2004.

[3] J. Gras, "Puzzle reassembly using model based reinforcement learning," 2019. [Online]. Available: https://johan-gras.github.io/projects/puzzlereassembly/

[4] V. Pammer et al., "Designing for engaging bci training: A jigsaw puzzle," in *the 2015 Annual Symposium on Computer-Human Interaction in Play*, 2015, pp. 667–672.

[5] L. Yang et al., "Jigsaw puzzle: Selective backdoor attack to subvert malware classifiers," in *IEEE Symposium on Security and Privacy (SP)*, 2023, pp. 719–736.

[6] A. R. Willis and D. B. Cooper, "Computational reconstruction of ancient artifacts," *IEEE Signal processing magazine*, vol. 25, no. 4, pp. 65–83, 2008.

[7] C. Doersch et al., "Unsupervised visual representation learning by context prediction," in *IEEE ICCV*, December 2015, pp. 1422–1430.

[8] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Computer Vision – ECCV 2016*, 2016, pp. 69–84.

[9] F. A. Andaló et al, "PSQP: Puzzle solving by quadratic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 2, pp. 385–396, 2017.

[10] ——, "Solving image puzzles with a simple quadratic programming formulation," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2012, pp. 63–70.

[11] F. A. Andaló and S. Goldenstein, "Computer vision methods applicable to forensic science," in *Workshop of Theses and Dissertations, Conference on Graphics, Patterns and Images (WTD/SIBGRAPI)*, 2013.

[12] F. A. Andaló et al, "Automatic reconstruction of ancient Portuguese tile panels," Instituto Nacional de Matemática Pura e Aplicada (IMPA), Tech. Rep. A773/2016, 2016.

[13] D. Sholomon et al., "An automatic solver for very large jigsaw puzzles using genetic algorithms," *Genetic Programming and Evolvable Machines*, vol. 17, no. 3, pp. 291–313, Sep 2016.

[14] M. Paumard et al., "Image reassembly combining deep learning and shortest path problem," in *Computer Vision – ECCV 2018*, 2018, pp. 155–169.

[15] ——, "Deepzzle: Solving visual jigsaw puzzles with deep learning and shortest path optimization," *IEEE Transactions on Image Processing*, vol. 29, pp. 3569–3581, 2020.

[16] ——, "Alphazzle: Jigsaw puzzle solver with deep monte-carlo tree search," *arXiv preprint*, 2023.

[17] C. Park et al., "Generating 2d lego compatible puzzles using reinforcement learning," *IEEE Access*, vol. 8, pp. 180 394–180 410, 2020.

[18] F. Jampy et al., "3d puzzle reconstruction for archeological fragments," in *Three-Dimensional Image Processing, Measurement (3DIPM), and Applications 2015*, vol. 9393. SPIE, 2015, pp. 56–64.

[19] M. Siam et al., "A comparative study of real-time semantic segmentation for autonomous driving," in *the IEEE CVPR workshops*, 2018, pp. 587–597.

[20] K. Khan et al., "Crowd counting using end-to-end semantic image segmentation," *Electronics*, vol. 10, no. 11, p. 1293, 2021.

[21] R. Miyamoto et al., "Vision-based road-following using results of semantic segmentation for autonomous navigation," in *IEEE ICCE-Berlin*, 2019, pp. 174–179.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE CVPR*, 2016, pp. 770–778.

[24] J. Deng et al., "Imagenet: A large-scale hierarchical image database," in *IEEE CVPR*, 2009, pp. 248–255.

[25] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, pp. 353–374, 2023.

[26] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, pp. 87–93, 2018.

[27] J. Wang et al., "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, oct 2021.

[28] J.-C. Chen, "Dijkstra's shortest path algorithm," *Journal of formalized mathematics*, vol. 15, no. 9, pp. 237–247, 2003.

[29] Ypsilantis et al, "The met dataset: Instance-level recognition for artworks," in *Neural Information Processing Systems Track on Datasets and Benchmarks*, vol. 1, 2021, pp. 1–12.

[30] R. Mottaghi et al, "The role of context for object detection and semantic segmentation in the wild," in *IEEE CVPR*, 2014, pp. 891–898.