Utilização de Métodos de Inteligência Artificial em Problemas de Otimização Topológica

João Vítor Omonte Neves (bolsista CNPq, RA 176423) e Renato Pavanello (orientador)
Departamento de Mecânica Computacional – Faculdade de Engenharia Mecânica – UNICAMP

Resumo

A área de *otimização topológica* estuda como distribuir espacialmente o material que constitui uma estrutura de modo a maximizar seu o desempenho, satisfazendo certas condições de contorno. O problema de reduzir o seu custo computacional ocupa uma posição de destaque nessa área. Este trabalho investiga a possibilidade de fazê-lo utilizando um método de inteligência artificial – o aprendizado de máquina (*machine learning*). A ideia fundamental é treinar redes neurais para prever a topologia otimizada por um método iterativo tradicional, a partir de dados de uma estrutura preliminarmente trabalhada pelo método. Em uma só etapa, isso é, sem as numerosas iterações dos métodos usuais, a rede treinada completaria a otimização, gerando a topologia final. O sucesso depende crucialmente da rede fazer previsões acuradas, mesmo em problemas diferentes daqueles usados no seu treinamento. Foram usadas duas redes convolucionais profundas, baseadas na U-Net, celebrizada na segmentação semântica de imagens médicas. Elas incorporam inovações conceituais importantes - o uso da *compliance* como input para as redes e o da entropia binária cruzada como função custo, sem os termos aditivos empregados na literatura. O teste escolhido foi o problema de minimização da *compliance* do *short cantilever*. Resolveram-se 48 variantes com o método iterativo *Bidirectional Evolutionary Structural Optimization* (BESO), mudando o ponto de aplicação da carga. Metade foi usada no treinamento e as demais para avaliar o desempenho das redes neurais. Além da redução prevista no tempo de processamento (54% - 94%), as redes geraram topologias muito semelhantes às do BESO [IoU = (95 ± 5) %]. O resultado está em linha com os na literatura, mas foi obtido com um número de dados de treinamento três ordens de magnitude menor. Dada a importância vital da quantidade de dados de treinamento no aprendizado de máquina em geral, é muito provável que os resultados possam ser substancialmente aprimorados. Destaca-se ainda que uma das redes neurais exibe uma capacidade singular de generalização quando utilizada em problemas radicalmente diferentes daqueles utilizados no treinamento. Em suma, os resultados indicam que o método de otimização topológica proposto neste trabalho é muito promissor.

Neste trabalho propomos um método de otimização topológica baseado na segmentação semântica com redes neurais profundas (AGGARWAL, 2018). O nosso ponto de partida é o trabalho pioneiro de Sosnovik e Oseledets (2017). Iremos comparar a otimização topológica de um corpo sob carregamento realizada pelo método que propomos com aquela feita com o tradicional método evolucionário BESO (HUANG; XIE, 2010). Primeiro, examina-se quão próximas as estruturas otimizadas com as redes neurais são daquelas geradas pelo BESO. Em seguida, investiga-se a capacidade das redes de resolver problemas radicalmente diferentes daqueles utilizados no treinamento. Por fim, estima-se a redução no tempo de processamento proporcionado pelo método proposto aqui.

Escolheu como problema-teste a otimização da estrutura *short cantilever*, visando a minimização da *compliance* total e a redução da quantidade de material à metade da inicial. A figura abaixo representa a estrutura em questão. Ela tem comprimento e largura de 50 mm, e o material tem módulo de Young E = 100 GPa e coeficiente de Poisson ν = 0,3. Foram resolvidos 48 problemas, correspondente à aplicação de uma força de 10.000 N, direcionada para baixo, em diferentes pontos no lado esquerdo, oposto ao engaste. A otimização BESO foi realizada com uma versão adaptada de um código em MATLAB, elaborado por Tainan Calixto e Renato Pavanello. Adotou-se ER=0,010, $AR_{máx}$ =0,05, $r_{mín}$ =3 mm e τ =0,01 % e N=5. Utilizou-se uma malha com 128x128 elementos finitos quadrados, elásticos, lineares e isotrópicos.

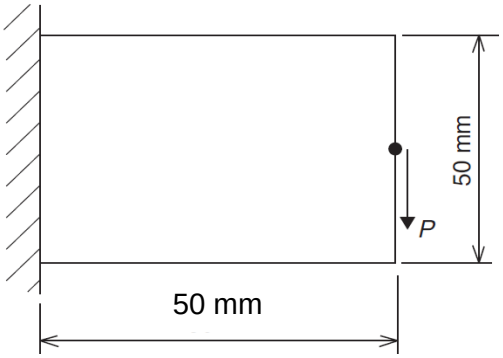

Figura 1 - *Short cantilever*, adaptada de Huang e Xie (2010).

Convém sintetizar o método de otimização, antes de tratar dos seus detalhes. As redes neurais (testaram-se duas variantes) recebem como *input* a distribuição da *compliance condicionada* (detalhes adiante) em uma estrutura parcialmente otimizada, por poucas iterações BESO. Em outras palavras, o método iterativo é interrompido após a remoção de uma quantidade de material bem menor que a

almejada. A distribuição da *compliance* nessa estrutura incipientemente otimizada é a base para o input das redes neurais. Os *outputs* das redes são as previsões das distribuições de material (topologias) finais produzidas pelo BESO ao atingir a remoção de material requerida. Trata-se um problema de segmentação semântica binária – a rede neural prevê a chance de cada elementos da estrutura ter ou não material, ao final da otimização. O treinamento das redes foi efetuado com 24 de exemplos, cada um composto pela *compliance* de uma estrutura parcialmente otimizada (*input* ou *feature*), e pela topologia final (*label* ou *ground truth*) obtida pelo BESO. A ideia geral é semelhante à de Sosnovik e Osoledets (2017), porém introduzimos as seguintes inovações importantes. 1) o *input* das redes é a *compliance condicionada* (vide adiante), não a topologia da estrutura parcialmente otimizada. 2) A função custo é a entropia cruzada binária, sem o termo adicional usado por Sosnovik. 3) Nossos dados de treinamento são gerados pelo BESO, não com o SIMP. 4) Introduzimos um processo de *filtragem* que transforma o *output* da rede com densidade contínua na topologia binária final.

Em um primeiro momento, o BESO foi interrompido quando ainda restava 70% da quantidade inicial de material. Como se mencionou, o objetivo do problema em questão é uma redução a 50%, com minimização da *compliance* total. Ao examinar a distribuição das *compliances* da estrutura parcialmente otimizada, percebe-se que em uma fração diminuta dos elementos essa grandeza tem um valor muito maior que nos demais – o valor máximo é cerca de 80 vezes maior que a média. Conjeturamos que possivelmente o comportamento da rede seria determinado pelos elementos com *compliance* maior. Assim, optou-se por *condicionar* (transformar) os dados de *input* para evitar esse comportamento. Utilizou-se o que denominamos a *compliance condicionada*, obtida com processos de *clipping* e *normalização*. O primeiro consiste em limitar as *compliances* a um valor x_L, igual a 1/30 do valor máximo no conjunto de dados de treinamento. Os valores maiores que esse limite são trocados por ele e os demais são mantidos. Em seguida, faz-se a normalização, multiplicando esses números por fator adequado, de maneira que os valores obtidos ocupam a faixa entre 0 e 1. Em resumo, sendo $\{x\}$ o conjunto das *compliances* calculadas com 70% de material, o *valor condicionado,* x_c , em um elemento onde a *compliance* vale *x,* é dado pela equação abaixo. Usando como entrada a *compliance condicionada*, as redes produzem como output matrizes de densidade com valores aproximadamente binários ($\rho \approx 0$ ou $\rho \approx 1$). Adotou-se como topologia final aquela obtida com o seguinte processo de *filtragem*. Os elementos com densidade maior que a mediana dos valores na estrutura teriam material (densidade filtrada $\rho_f = 1$), enquanto os outros estariam vazios ($\rho_f = 0$).

$$x_c(x) = \begin{cases} x/x_L \ , \ x \le x_L \\ 1 \ , \ x > x_L \end{cases} \quad \text{, onde } x_L = \frac{max\ \{x\}}{30}$$

A figura abaixo sintetiza o processo de otimização delineado. À esquerda, entram na rede neural as *compliances condicionadas*, mostradas na figura em uma escala de tons de cinza. A saída da rede, à sua direita, é a distribuição da densidade prevista (tons de cinza). Em seguida, esses números são convertidos pela filtragem em uma densidade binária: $\rho_f = 0 \rightarrow$ sem material e $\rho_f = 1 \rightarrow$ material presente.

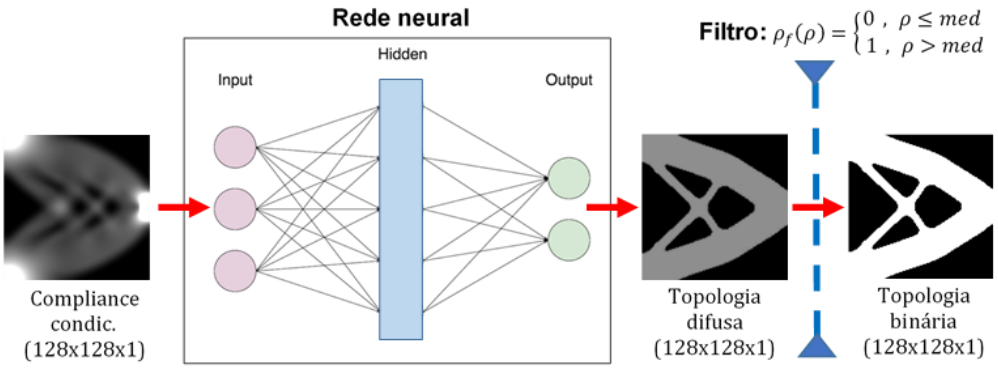

Figura 2 – Representação esquemática do processo de otimização topológica. A rede neural recebe a distribuição das *compliances condicionadas*, retorna uma topologia com densidade contínua, que a filtragem transforma na topologia binária.

As duas redes usadas são inspiradas na célebre U-Net (RONNEBERGER; FISCHER; BROX, 2015). Uma característica atrativa desse modelo é a ausência de camadas densamente conectadas. Isso assegura que a rede treinada, com exemplos com uma dada resolução, seja também apta a fazer previsões a partir de *inputs* com resolução diferente, sem que seja necessário um novo treinamento. A rede que nomeamos como R1 foi criada em Python 3.7, usando a biblioteca TensorFlow 2.0, enquanto a sua sucessora, R2, foi implantada com o programa Mathematica 12.1. O código mais compacto de criação da segunda é

apresentado no apêndice. A diferença relevante na R1 é a substituição das camadas de *normalização de batch* por camadas de *dropout* de 30%. Essas são utilizadas para reduzir problemas com *overfitting*. Para treinar as redes foram usadas as soluções de 24 dos 48 problemas de otimização do *short cantilever*, obtidas com o BESO. Como foi dito, cada exemplo de treinamento consistia da *compliance condicionada* (*input*), calculada quando o BESO atingia 70% do volume inicial, e da topologia final da otimização (*label*). As outras 24 soluções do BESO foram usadas como teste de validação do processo proposto. A rede R1 foi treinada com os dados distribuídos em *mini-batches* com 4 exemplos, por 100 épocas, utilizando o método ADAM. No caso da rede R2, foram usados *mini-batches* de tamanho 8. No treinamento de R1 e de R2, a função custo foi a entropia cruzada binária, definida pela expressão abaixo, onde $\boldsymbol{x_{lab}}$ e $\boldsymbol{x_{pre}}$ são respectivamente as matrizes de densidade no final da otimização BESO (label) e aquela prevista pela rede.

$$C(\boldsymbol{x_{lab}}, \boldsymbol{x_{pre}}) = -\frac{1}{128^2}\sum_{i=1}^{128}\sum_{j=1}^{128}\left[x_{lab}^{ij}\ln x_{pre}^{ij} + \left(1 - x_{lab}^{ij}\right) ln\left(1 - x_{pre}^{ij}\right)\right]$$

A figura abaixo retrata as topologias otimizadas com a rede R2 aplicada aos dados de validação. Cada par corresponde a um problema de otimização do *short cantilever,* com uma força aplicada em ponto diferente no lado direito da estrutura e direcionada para o pé da folha. Em cada par, a figura da esquerda é a *ground truth* (*label), a* topologia final obtida com o BESO, e a da direita é a previsão feita a partir da *compliance condicionada*, calculada quando havia 70% do material inicial. Nota-se que apenas em poucos casos a estrutura prevista é consideravelmente diferente da esperada, por vezes com partes incompletas, sem sentido do ponto de vista estrutural. É importante lembrar que os problemas tratados aqui não fizeram parte do treinamento, e que esse foi realizado com apenas 24 exemplos. No trabalho de Sosnovik e Osoledets (2017), o treinamento foi realizado com 40.000 exemplos. As topologias previstas para os problemas de treinamento (não foram mostradas) são todas visualmente indistinguíveis dos seus *labels*.

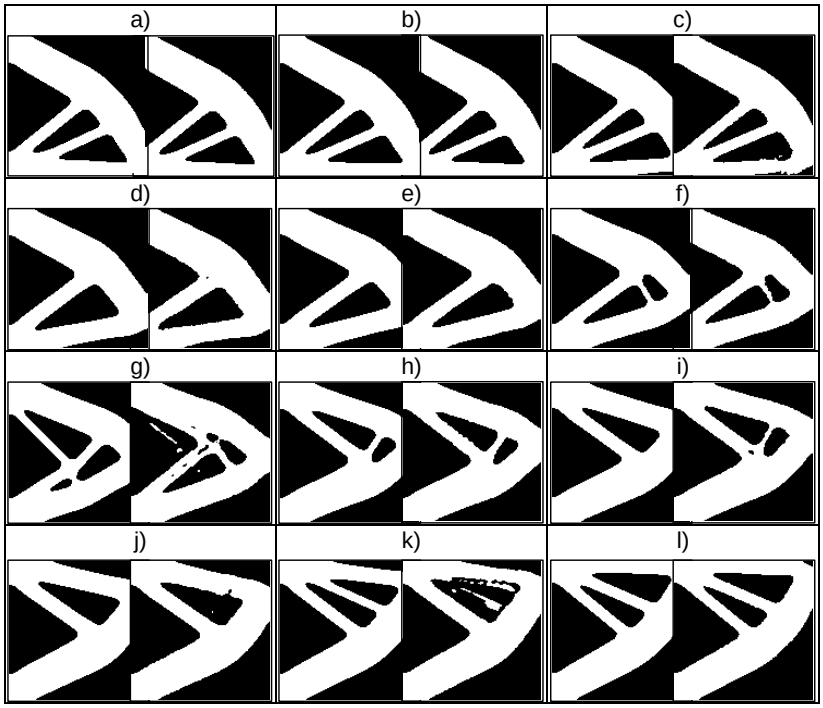

Figura 3 - Pares com a topologia final obtida com o BESO (esquerda) e aquela prevista (direita) com o processo proposto, usando a rede R2 e os dados de validação.

A acurácia binária e principalmente a *intersection over union* (IoU) são as métricas usuais para quantificar quão próximas são as topologias correta e as previstas. No caso presente, onde não há desbalanço entre as classes (o número de elementos com e sem material é igual), as duas métrica são adequadas. Considerando o caráter estocástico do treinamento da rede neural, repetiu-se o treinamento e a otimização cinco vezes para se estabelecer a sua reprodutibilidade. A tabela abaixo traz os valores médios obtidos com os dados de validação e de treinamento, com as redes R1 e R2. As duas preveem corretamente se há ou não material em mais de 96% dos elementos. Esses resultados são poucos pontos percentuais abaixo dos reportados por Sosnovik e Oseledets (2017).

Tabela 1 - IoU e acurácia média com desvio médio (em porcentagem) para os dados de treinamento e de validação com as redes R1 e R2.

	R1-valid.	R1-treino	R2-valid.	R2-treino
IoU (%)	93,4 ± 0,2	97,7 ± 0,4	95 ± 5	99,98 ± 0,04
Acu. bin. (%)	96,5 ± 0,2	98,9 ± 0,2	97 ± 3	99,99 ± 0,02

Os resultados apresentados até aqui foram obtidos a partir dos dados de *compliance condicionada*, calculados no momento em que a otimização BESO chegava a 70% da quantidade inicial de material. Foi investigado se a rede poderia ainda fazer boas previsões, caso fizesse uma parte maior do trabalho do BESO. Mais especificamente, examinou-se quão precisas são as previsões da topologia final, se a rede é treinada com os dados de *compliance* obtidos quando a quantidade de material ainda era 80%, 90%, e mesmo 100% da original. A experiência foi realizada com a rede R1. Observou-se que a IoU com os dados validação oscilou entre 90,5 e 93,5%. Apesar da queda não ser grande, a análise visual das estruturas revela que ela está relacionada com otimizações piores.

Como foi mencionado, a aplicabilidade das redes neurais depende da sua capacidade de resolver problemas diferentes daqueles usados no treinamento. Nas seções anteriores, viu-se que as redes R1 e R2 tiveram um desempenho promissor nessa tarefa. Entretanto, como no trabalho de Sosnovik e Osoledets (2017), os nossos problemas de validação eram parecidos com os de treinamento. É pertinente (e interessante) testar as redes em problemas radicalmente diferentes daqueles usados no seu treinamento, como se discute na sequência. A figura abaixo mostra as topologias finais otimizadas pelo BESO (partes a, b e c) e as prevista pelo nosso método (a', b', c') com a rede R2, em problemas onde a força (seta vermelha) foi aplicada no lado superior, ou no lado esquerdo da estrutura. Vale frisar que em todos os exemplos de treinamento, a força foi aplicada no lado direito. As três topologias na parte superior da figura são localmente completamente diferentes daquelas nos dados de treinamento (semelhantes às da figura 3). Naturalmente uma pessoa perceberá uma relação de simetria, entretanto essa noção não foi introduzida no método com a rede neural. Contudo, surpreendente para o autor, a rede aprendeu a ideia. Curiosamente a rede R2 tem uma capacidade de generalização consideravelmente maior que a R1. Seria importante investigar futuramente que característica da arquitetura da rede é responsável por essa habilidade.

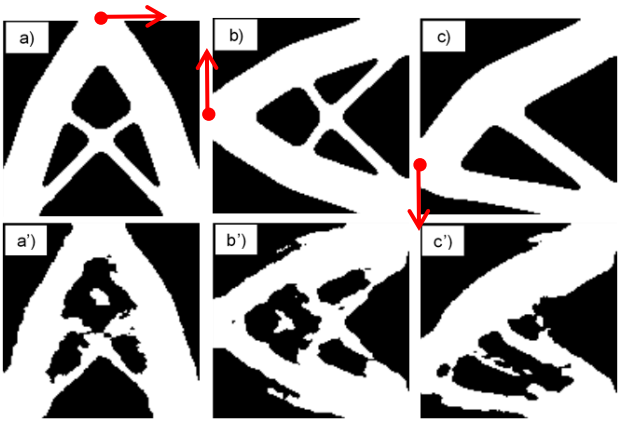

Figura 4 – a, b e c mostram a topologia final otimizada pelo BESO (ground truth) com a força e seu ponto de aplicação indicados em vermelho (o engaste é no oposto). A parte debaixo (a', b' e c') traz as previsões obtidas com R2.

A redução do tempo de processamento computacional é a motivação principal para o uso do aprendizado de máquina em problemas de otimização topológica. O treinamento de uma rede neural é uma tarefa computacional pesada. No entanto, uma vez treinada, a rede pode ser muito mais rápida, já que o processamento do input é realizado em uma só etapa, sem processos iterativos. Segue uma análise da redução no tempo de processamento proporcionado pelas redes. Adianta-se que os tempos mencionados abaixo foram obtidos com o mesmo hardware. O tempo médio demandado por uma otimização BESO completa (remoção de 50% de material) é de 37,5 s. A rede neural leva pouco menos de 0,1 s para fazer a previsão da topologia otimizada. Entretanto, é preciso considerar o tempo necessário para que o BESO realize a otimização preliminar que alimenta a rede. Para chegar a 70, 80, 90 e 100% da quantidade inicial de material o BESO leva 17,3 s, 11,1 s, 5,7 s e 2,1 s, respectivamente. Nota: o último valor (*redução a 100%*) é o tempo que o programa leva para calcular a *compliance* na estrutura inicial por elementos finitos. Assim, se usarmos o método proposto neste trabalho, partindo dos dados da estrutura parcialmente otimizada com 70% do material inicial, a redução de tempo com relação a otimização completa com o BESO seria de 37,5 s - 17,3 s – 0,1 s = 20,1 s. De fato, gastaríamos 17,3 s para gerar os dados necessários para a rede, mas a partir daí, ela completaria em 0,1 s a otimização que com o BESO leva 37,5 s. O método com a rede neural reduz o tempo de processamento em 53,6%. Se partimos da estrutura sem otimização (100% do material inicial), a redução seria de 93,6%. É provável que os ganhos sejam ainda maiores no caso de problemas de otimização de estruturas com maior resolução. É interessante mencionar que o tempo médio de treinamento da rede R1 com os 24 exemplos é de (21±1) ×10 s. Todavia, como argumentou-se, esse tempo é irrelevante na avaliação da redução do custo computacional proporcionado pela rede neural treinada.

Conclui-se que o método proposto neste trabalho é um caminho muito promissor no esforço atual para reduzir o custo computacional da otimização topológica. As redes neurais foram capazes de prever acuradamente as topologias otimizadas a partir de dados de uma otimização preliminar. Ademais, os resultados revelam que o método proposto tem um grande potencial de generalização para tratar de problemas de otimização radicalmente diferentes daqueles utilizados no treinamento das redes neurais. É importante destacar que neste trabalho exploratório utilizou-se um número muito reduzido de exemplos de treinamento (24) para as redes e que não se empreendeu a otimização dos seus hiperparâmetros (taxa de *dropout*, *stride*, tamanho dos filtros, etc) – foram simplesmente adotados valores usuais em outros problemas. Na mesma linha, considerando a importância capital do número de exemplos de treinamento no aprendizado de máquina, é muito provável que a simples ampliação dessa massa de dados proporcione um aumento substancial na qualidade dos resultados do método proposto. Ainda não é possível comparar objetivamente os resultados obtidos aqui com a referência maior na literatura - o trabalho de Sosnovik e Osoledets (2017). Os autores trataram da otimização de uma estrutura diferente. Com essa ressalva, notamos que em termos da precisão das topologias previstas, os nossos resultados estão apenas ligeiramente aquém dos apresentados pelos autores. No entanto, Sosnovik e Osoledets utilizaram 40.000 exemplos de treinamento. É de grande importância comparar futuramente os resultados dos dois métodos no tratamento de um mesmo problema, usando os mesmos dados de treinamento.

Por fim, destacamos que este trabalho traz as seguintes inovações conceituais importantes. 1) O *input* para a rede neural é baseado na *compliance* de uma estrutura preliminarmente otimizada. 2) A redução na quantidade de material prescrita na formulação do problema é obtida por um processo de *filtragem*, não pela introdução de um termo aditivo na função custo, como fez Sosnovik. 3) Adotou-se como função custo a entropia cruzada binária. Os últimos dois pontos permitem que a rede neural busque unicamente a minimização da *compliance*, sem ter que simultaneamente trabalhar para atingir uma determinada quantidade e material.

Referências bibliográficas

HUANG, X.; XIE, Y.M.. **Evolutionary topology optimization of continuum structures:** Methods and aplications. West Sussex: John Wiley And Sons Ltd, 2010. 223 p.

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas, U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv:1505.04597 [cs], 2015.

SOSNOVIK, Ivan; OSELEDETS, Ivan. Neural networks for topology optimization. arXiv:1709.09578 [cs, math], 2017. Disponível em: <http://arxiv.org/abs/1709.09578>. Acesso em: 13 jun. 2019.

Apêndice

Segue-se o código para criação da rede neural R2 no programa Wolfram Mathematica, versão 12.1.

Definição da rede R2 para dados 128 x 128 pixels:

```
conv[n_] := NetChain[{ConvolutionLayer[n, 3, "PaddingSize" → {1, 1}], Ramp, BatchNormalizationLayer[],
    ConvolutionLayer[n, 3, "PaddingSize" → {1, 1}], Ramp, BatchNormalizationLayer[]}]

pool := PoolingLayer[{2, 2}, 2]

dec[n_] := NetGraph[{"deconv" → DeconvolutionLayer[n, {2, 2}, "Stride" → {2, 2}], "cat" → CatenateLayer[], "conv" → conv[n]},
   {NetPort["Input1"] → "cat", NetPort["Input2"] → "deconv" → "cat" → "conv"}];

r2 := NetGraph[<|"enc_1" → conv[64], "enc_2" → {pool, conv[128]}, "enc_3" → {pool, conv[256]},
   "enc_4" → {pool, conv[512]}, "enc_5" → {pool, conv[1024]}, "dec_1" → dec[512], "dec_2" → dec[256],
   "dec_3" → dec[128], "dec_4" → dec[64], "map" → {ConvolutionLayer[1, {1, 1}], LogisticSigmoid}|>,
  {NetPort["Input"] → "enc_1" → "enc_2" → "enc_3" → "enc_4" → "enc_5", {"enc_4", "enc_5"} → "dec_1",
   {"enc_3", "dec_1"} → "dec_2", {"enc_2", "dec_2"} → "dec_3", {"enc_1", "dec_3"} → "dec_4", "dec_4" → "map"},
  "Input" → {1, 128, 128}]

r2
```