



APRENDIZEGEM PROFUNDA APLICADA

Assignment 2

Autor:

Gonçalo Bastos
Leonardo Cordeiro

Numero de Estudante:

2020238997
2020228071

Novembro 26, 2024

1 Introdução

Este relatório apresenta o desenvolvimento e análise de modelos de deep learning no contexto de duas tarefas distintas: reconstrução de imagens e detecção de objetos. O trabalho está dividido em duas partes principais.

Na primeira parte, foram implementados, treinados e avaliados três modelos diferentes de Autoencoders—Autoencoder (AE), Variational Autoencoder (VAE) e Denoising Autoencoder (DAE)—usando o dataset FashionMNIST. Estes modelos foram comparados com base no desempenho de reconstrução, visualização do espaço latente e precisão de classificação utilizando as características aprendidas pelo *encoder*.

Na segunda parte, foi treinada e avaliada uma rede de deteção de objetos YOLOv5 utilizando um subconjunto do dataset KITTI. O estudo inclui uma comparação entre treinar o modelo de raiz (*from scratch*) e utilizar *transfer learning* com pesos pré-treinados para avaliar o desempenho e as capacidades de generalização.

2 Part I - Reconstrução e remoção de ruído com Autoencoders

2.1 Análise Teórica: AE, VAE e DAE

Os modelos AE, VAE e DAE utilizam um *encoder* para capturar as características mais relevantes das imagens de entrada, comprimindo-as em um vetor unidimensional denominado *bottleneck*. Este vetor representa o espaço latente e serve como base para a descodificação realizada pelo *decoder*. Enquanto AE e DAE utilizam diretamente este vetor no processo de reconstrução, o VAE introduz etapas intermediárias que serão descritas a seguir.

No caso do **AE (Autoencoder)**, o vetor no *bottleneck* é processado pelo *decoder* para reconstruir a imagem original, aplicando operações inversas às realizadas durante a codificação. O treino do modelo é baseado na minimização de uma função de perda de reconstrução, que compara a entrada e a saída. As funções mais comuns são o *Mean Squared Error* (MSE), que mede a distância entre os valores previstos e reais dos píxeis, e o *Binary Cross-Entropy* (BCE), que avalia a probabilidade logarítmica das previsões. Como resultado, as reconstruções do AE são geralmente muito fíéis à entrada, preservando detalhes com alta precisão.

O **VAE (Variational Autoencoder)** amplia o conceito do AE ao gerar dois vetores no *bottleneck*: a média (μ) e a variância logarítmica ($\log \sigma^2$). Esses vetores definem uma distribuição probabilística das características extraídas das imagens. Para isso, utiliza-se o *reparameterization trick*, que permite amostrar um vetor latente z a partir da distribuição calculada ($z = \mu + \sigma \odot \epsilon$, onde $\epsilon \sim \mathcal{N}(0, 1)$). O vetor z é então passado ao *decoder* para reconstrução. Além da perda de reconstrução, o treino do VAE utiliza a *KL Divergence Loss* (KLD), que regulariza o espaço latente, aproximando a distribuição aprendida de uma distribuição normal padrão. Comparado ao AE, as saídas do VAE são mais suaves e menos detalhadas devido à sua abordagem probabilística, resultando em edges menos nítidos e texturas mais homogêneas.

O **DAE (Denoising Autoencoder)** utiliza a mesma arquitetura do AE, mas é treinado para reconstruir imagens limpas a partir de entradas corrompidas. Durante o treino, random noise é adicionado às imagens de entrada, degradando os valores dos píxeis. O modelo aprende a eliminar este ruído minimizando a perda de reconstrução, normalmente baseada no MSE. Este método torna o DAE mais robusto a ruídos nos dados, permitindo focar em características significativas. As saídas do DAE mostram reconstruções precisas das imagens limpas, mesmo quando as entradas apresentam altos níveis de corrupção.

Resumo Comparativo:

- **AE:** Focado na reconstrução com alta fidelidade das imagens de entrada, mas sensível a ruídos.
- **VAE:** Produz saídas mais suaves e variadas devido à representação probabilística, mas com menor nível de detalhe.
- **DAE:** Especializado em recuperar imagens limpas a partir de entradas ruidosas, e com a característica de ser robusto a perturbações.

2.2 Arquiteturas

Nesta seção, apresentamos as arquiteturas desenvolvidas para os modelos AE, VAE e DAE, juntamente com uma descrição detalhada das suas diferenças e características específicas. As tabelas e diagramas a seguir sintetizam as estruturas e funções de cada modelo.

2.2.1 Autoencoder (AE)

O **AE (Autoencoder)** foi implementado com uma arquitetura baseada em convoluções para o *encoder* e *decoder*. O *encoder* reduz progressivamente a dimensão espacial das imagens de entrada (28×28) para uma representação

comprimida (*bottleneck*) de $3 \times 3 \times 128$. O *decoder* reconstrói a imagem a partir do *bottleneck*, utilizando camadas *ConvTranspose2D* e *Upsampling*. A função de perda utilizada foi o *Mean Squared Error* (MSE).

| Camada | Operação | Dimensão de Saída |
|-----------|--------------------------------|-------------------------|
| Entrada | Imagen $28 \times 28 \times 1$ | $1 \times 28 \times 28$ |
| Encoder 1 | Conv2D (8 filtros, stride=2) | $8 \times 14 \times 14$ |
| Encoder 2 | Conv2D (8 filtros, stride=1) | $8 \times 14 \times 14$ |
| Encoder 3 | Conv2D (16 filtros, stride=3) | $16 \times 5 \times 5$ |
| Encoder 4 | Conv2D (32 filtros, stride=1) | $32 \times 5 \times 5$ |
| Encoder 5 | Conv2D (128 filtros, stride=2) | $128 \times 3 \times 3$ |
| Decoder 1 | ConvTranspose2D (stride=2) | $32 \times 5 \times 5$ |
| Decoder 2 | Conv2D (stride=1) | $16 \times 5 \times 5$ |
| Decoder 3 | ConvTranspose2D (stride=3) | $8 \times 14 \times 14$ |
| Decoder 4 | Conv2D (stride=1) | $8 \times 14 \times 14$ |
| Decoder 5 | Upsample para 28×28 | $1 \times 28 \times 28$ |
| Saída | Sigmoid | $1 \times 28 \times 28$ |

Table 1: Arquitetura do Autoencoder (AE).

2.2.2 Variational Autoencoder (VAE)

O **VAE (Variational Autoencoder)** expande o AE ao introduzir dois vetores no *bottleneck*: média (μ) e variância logarítmica ($\log \sigma^2$). Utilizando o *reparameterization trick*, o modelo gera um vetor z amostrado a partir da distribuição latente, que é posteriormente decodificado. O VAE adiciona a *KL Divergence Loss* à função de perda para regularizar o espaço latente.

| Camada | Operação | Dimensão de Saída |
|------------------|--|-------------------------|
| Entrada | Imagen $28 \times 28 \times 1$ | $1 \times 28 \times 28$ |
| Encoder 1 | Conv2D (8 filtros, stride=2) | $8 \times 14 \times 14$ |
| Encoder 2 | Conv2D (16 filtros, stride=2) | $16 \times 7 \times 7$ |
| Encoder 3 | Conv2D (32 filtros, stride=2) | $32 \times 4 \times 4$ |
| Bottleneck | Linear (μ e $\log \sigma^2$) | 20 |
| Reparametrização | $z = \mu + \sigma \cdot \epsilon$ | 20 |
| Decoder 1 | Linear para $32 \times 4 \times 4$ | $32 \times 4 \times 4$ |
| Decoder 2 | ConvTranspose2D (stride=2) | $16 \times 7 \times 7$ |
| Decoder 3 | ConvTranspose2D (stride=2, output_padding=1) | $8 \times 14 \times 14$ |
| Decoder 4 | ConvTranspose2D (stride=2, output_padding=1) | $1 \times 28 \times 28$ |
| Saída | Sigmoid | $1 \times 28 \times 28$ |

Table 2: Arquitetura do Variational Autoencoder (VAE).

2.2.3 Denoising Autoencoder (DAE)

O **DAE (Denoising Autoencoder)** utiliza a mesma arquitetura do AE, mas é treinado com entradas ruidosas. O ruído é adicionado às imagens durante o treino, e o modelo aprende a reconstruir as imagens originais limpas, minimizando a perda de reconstrução (MSE). Esta abordagem melhora a robustez do modelo em condições de dados com ruído.

- **Adição de ruído:**

$$\text{noisy_data} = \text{data} + \text{noise_factor} \cdot \text{torch.randn_like}(\text{data})$$

- **Perda:** Apenas *Mean Squared Error* (MSE).

- **Arquitetura:** Igual ao AE.

2.2.4 Resumo Comparativo

Principais diferenças assinaladas na seguinte tabela:

| Características | AE | VAE | DAE |
|------------------|----------------|---------------------|----------------|
| Objetivo | Reconstrução | Probabilística | Denoising |
| Ruído | Não | Não | Sim |
| Função de Custo | MSE | MSE + KL Divergence | MSE |
| Latent Space | Determinístico | Probabilístico | Determinístico |
| Robustez a Ruído | Fraca | Média | Elevada |

Table 3: Resumo comparativo entre AE, VAE e DAE.

2.3 Resultados Obtidos com os Modelos

Nesta seção, reportamos os resultados obtidos com os modelos desenvolvidos, incluindo os hiperparâmetros utilizados, as curvas de perda (*loss curves*), os pares de entrada e reconstrução, o erro médio de reconstrução e as visualizações do espaço latente (*latent space*) através de t-SNE.

2.3.1 Hiperparâmetros e Métricas

Os hiperparâmetros e as métricas de treino e validação para os três modelos (AE, VAE e DAE) estão compilados na Tabela 4.

Table 4: Hiperparâmetros e Métricas de Treino e Validação dos Modelos.

| Modelo | Optimizer | Learning Rate | Loss Function | Train Loss | Validation Loss | Latent Dim. |
|--------|-----------|--------------------|---------------------|------------|-----------------|-------------|
| AE | Adam | 1×10^{-3} | MSE | 0.0057 | 0.0058 | N/A |
| VAE | Adam | 1×10^{-3} | MSE + KL Divergence | 20.4469 | 5.1369 | 20 |
| DAE | Adam | 1×10^{-3} | MSE | 0.0140 | 0.0141 | N/A |

2.3.2 Curvas de Perda

As Figuras 1 apresentam as curvas de perda de treino e validação dos três modelos. Os gráficos estão organizados lado a lado para facilitar a comparação.

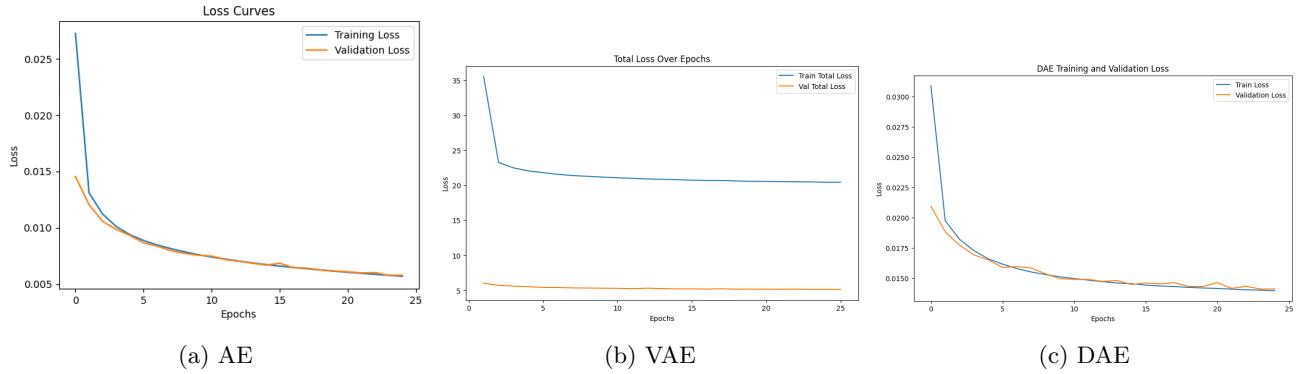


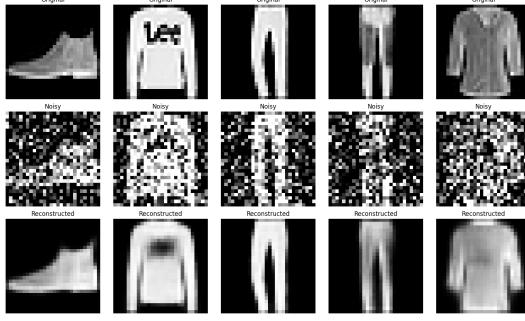
Figure 1: Curvas de Perda (Treino e Validação) dos Modelos AE, VAE e DAE.

2.3.3 Pares de Entrada e Reconstrução

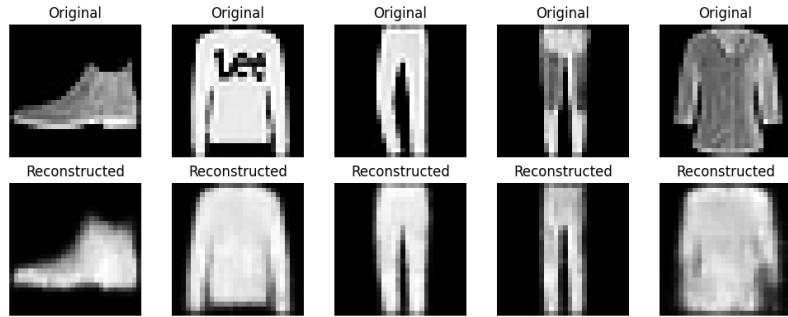
As Figuras 2 mostram os pares de entrada (originais e com ruído) e reconstrução dos três modelos. Cada linha representa um modelo específico.

2.3.4 Distribuição dos Erros de Reconstrução

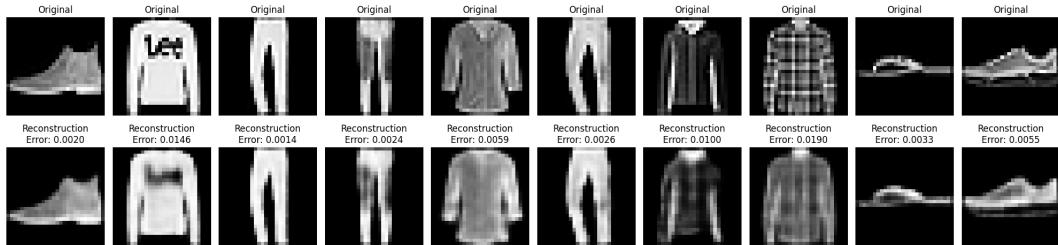
As distribuições dos erros de reconstrução estão apresentadas na Figura 3, permitindo observar as diferenças na performance dos modelos.



(a) DAE



(b) VAE



(c) AE

Figure 2: Pares de Entrada (Linha Superior) e Reconstrução (Linha Inferior) para os Modelos AE, VAE e DAE.

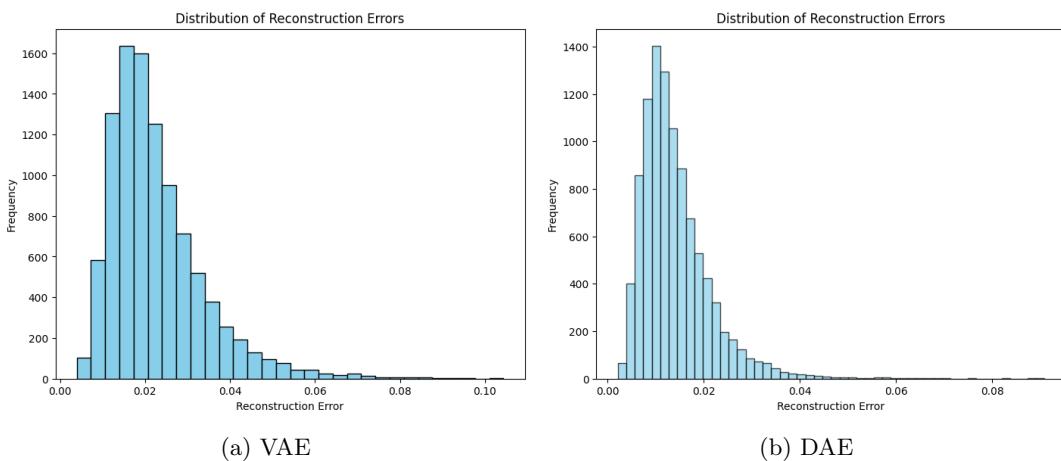


Figure 3: Distribuições dos Erros de Reconstrução dos Modelos AE, VAE e DAE.

2.3.5 Visualização do Espaço Latente

A Figura 4 apresenta as projeções t-SNE do espaço latente dos três modelos.

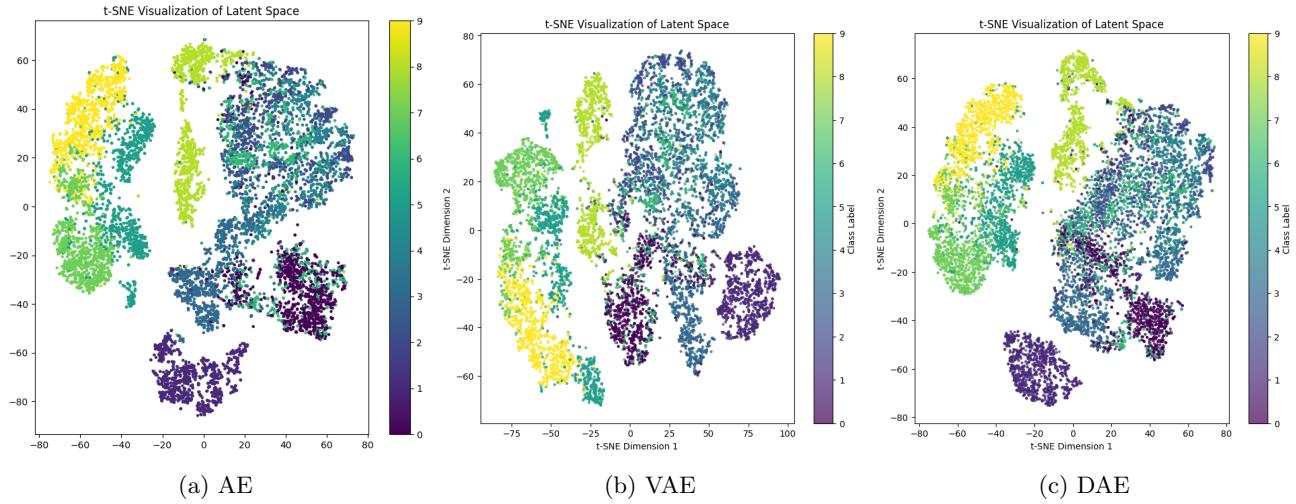


Figure 4: Visualização t-SNE do Espaço Latente para os Modelos AE, VAE e DAE.

2.3.6 Análise Comparativa dos Resultados

Os resultados obtidos com os três modelos — AE, VAE e DAE — apresentaram características distintas e coerentes com o esperado, evidenciando os pontos fortes e limitações de cada abordagem:

Análise do Autoencoder (AE): Os resultados obtidos com o Autoencoder evidenciam uma capacidade robusta para reconstruir as imagens do dataset FashionMNIST. As reconstruções apresentam uma elevada fidelidade, preservando as principais características das imagens de entrada. Contudo, quando testado com imagens contendo ruído, o modelo reproduz o ruído na saída, o que demonstra que o AE não foi treinado para ignorar elementos indesejados no *input*. As curvas de perda mostram uma convergência estável, com diferenças mínimas entre as perdas de treino e validação, indicando que o modelo não sofreu de sobreajuste (*overfitting*).

A análise do espaço latente através da visualização t-SNE destaca uma separação clara entre as classes, sugerindo que o *encoder* conseguiu aprender uma representação latente eficaz dos dados. No entanto, é importante notar que o AE apresenta sensibilidade a ruído, uma limitação que será explorada na comparação com os modelos VAE e DAE.

Análise do Variational Autoencoder (VAE): Como previsto, o VAE produz imagens onde as linhas aparecem menos definidas na reconstrução e a cor do objeto original surge com uma intensidade mais uniforme. Ao ser testado com imagens ruidosas, o modelo consegue remover uma parte significativa do ruído, mas demonstra dificuldade em reconstruir detalhes mais finos, como o logotípido presente numa camisola.

O VAE revelou maior diversidade nas reconstruções, embora com menor precisão em comparação ao Autoencoder. A introdução da *KL Divergence Loss* desempenhou um papel importante na regularização do espaço latente, permitindo obter representações latentes mais organizadas, como se observa na visualização t-SNE. No entanto, os erros de reconstrução foram mais elevados devido à natureza probabilística do modelo, algo que é esperado em VAEs.

Análise do Denoising Autoencoder (DAE): Os resultados obtidos mostram que o DAE foi capaz de reconstruir as imagens originais com sucesso, mesmo quando estas continham um nível significativo de ruído no *input*. O modelo demonstrou uma capacidade superior de filtrar o ruído e preservar os detalhes relevantes das imagens. A análise da distribuição dos erros de reconstrução revela que a maioria das imagens reconstruídas possui erros baixos, evidenciando a eficácia do modelo.

Na análise do *Latent Space*, observa-se que as classes estão mais afastadas entre si, confirmando a capacidade do *encoder* de aprender representações discriminativas mesmo em condições ruidosas. O treino com imagens contendo ruído permitiu que o modelo extraísse características mais robustas. Estes resultados reforçam a robustez do DAE para cenários onde os dados de entrada podem estar degradados.

Comparação Geral: Os três modelos obtiveram os resultados esperados, cada um destacando-se em diferentes cenários:

- **AE e DAE:** Devolveram as reconstruções mais exatas das imagens sem ruído, com o AE mostrando uma alta fidelidade na reconstrução de detalhes das imagens originais.
- **VAE:** Produziu reconstruções com margens mais atenuadas e intensidade uniforme nos pixels, uma característica esperada da teoria probabilística subjacente ao modelo.
- **DAE:** Demonstrou ser o mais eficaz dos três na reconstrução de imagens distorcidas por ruído, uma vez que foi treinado especificamente para este cenário. O DAE conseguiu remover o ruído mantendo as características relevantes da imagem original.

Quando comparados em cenários de entrada ruidosa:

- **AE:** Tenta reconstruir a imagem original na íntegra, reproduzindo o ruído no *output*, o que compromete a qualidade da reconstrução.
- **VAE:** Remove parte do ruído, mas distorce a imagem ao ponto de, em alguns casos, reconstruir o dígito errado. Isso pode ser explicado pela natureza probabilística do modelo, onde características alteradas pelo ruído podem aproximar o dígito de outra classe no espaço latente.
- **DAE:** Supera os demais ao filtrar o ruído com maior eficácia e preservar os detalhes principais, demonstrando robustez superior em condições adversas.

Concluímos que o DAE é a solução ideal em cenários onde os dados estão sujeitos a degradação, enquanto o AE e o VAE podem ser úteis em situações onde não há ruído ou onde se deseja uma representação latente mais regularizada. A combinação de modelos ou ajustes específicos poderia melhorar ainda mais os resultados para aplicações práticas.

2.4 Modelos de Classificação

Inicialmente o encoder dos arquiteturas AE, VAE e DAE treinado de forma não supervisionada, reconstrução das imagens de entrada. Isto permite ao encoder aprender representações do espaço latente, capturando as características dos dados como cantos e texturas. Quando o contexto muda para a classificação, o encoder funciona como um reconhecedor de características. Em vez de aprender as representações do 0, o classificador usa diretamente as features pré-treinadas.

Freezing the encoder – permite que a representação do espaço latente aprendida não seja modificada durante a classificação supervisionada. Isto permite estabilidade ao modelo, previne o modelo de realizar overfitting e ainda reduz o custo computacional. Isto é feito com o código `with torch.no_grad()`.

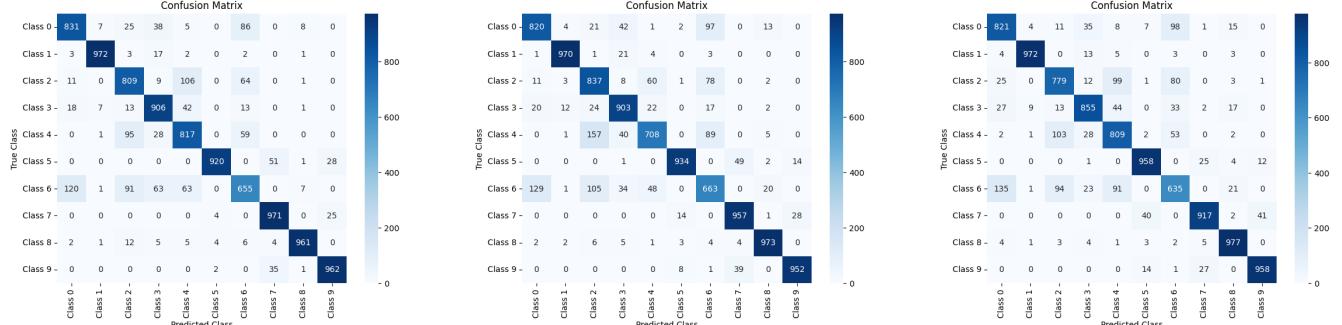
Adição de camadas de classificação – permite mudar o propósito do encoder para classificação. Adicionamos fully connected layers para mapear a representação do estado latente em probabilidades para as classes. Isto inclui:

- **Flattening Layer:** Converte o encoder multi-dimensional em um vetor.
- **Fully Connected Layers:** Desempenham a classificação (Dense Layers + Funções de ativação ReLU).
- **Output Layer:** Mapeia a dense layer para o número de classes do dataset.

Table 5: Resultados de Classificação AE, VAE, and DAE Modelos

| Modelo | Accuracy (%) | Precision (%) | Recall (%) | F1 Score (%) |
|--------------|--------------|---------------|------------|--------------|
| AE_Classify | 88.04 | 88.06 | 88.04 | 87.98 |
| VAE_Classify | 87.17 | 87.27 | 87.17 | 87.11 |
| DAE_Classify | 86.81 | 86.70 | 86.81 | 86.72 |

Os resultados da classificação revelam que o AutoEncoder superou tanto o Variational Autoencoder como o Denoising Autoencoder em todas as métricas de avaliação. O AE alcançou a maior accuracy de 88.04%, precisão de 88.06%, recall 88.04% e F1 score de 87.98%, demonstrando a sua capacidade para capturar características determinísticas e específicas no espaço latente, tornando altamente eficiente para classificação. VAE com uma accuracy de 87% e F1 score de 87%, exibindo uma classificação um pouco inferior devido à natureza probabilística do espaço latente, o que introduz alguma variabilidade intrínseca. Contudo o DAE alcançou uma accuracy mais baixa 86%. Concluindo, o AE com seu espaço latente determinístico tem um melhor trade-off entre precision e recall, enquanto o VAE e o DAE mostram potencial em tratar dados mais complexos ou com ruído.

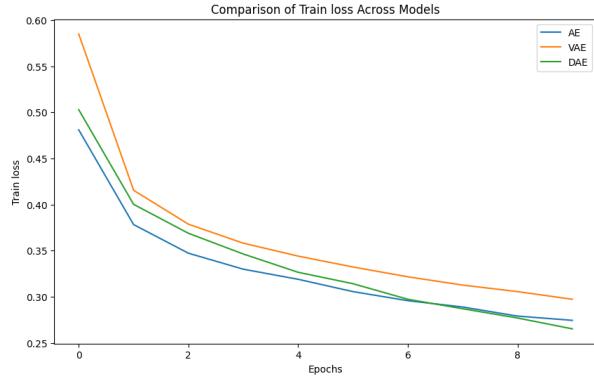


(a) Confusion Matrix do AE

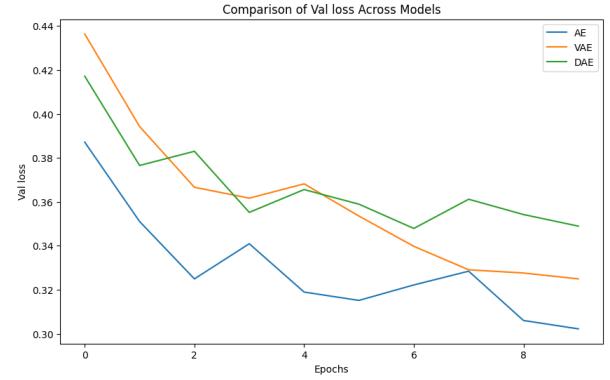
(b) Confusion Matrix do VAE

(c) Confusion Matrix do DAE

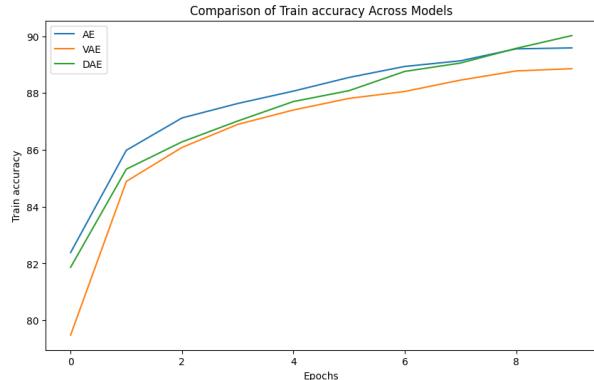
Figure 5: Confusion Matrices dos modelos: AE, VAE, e DAE



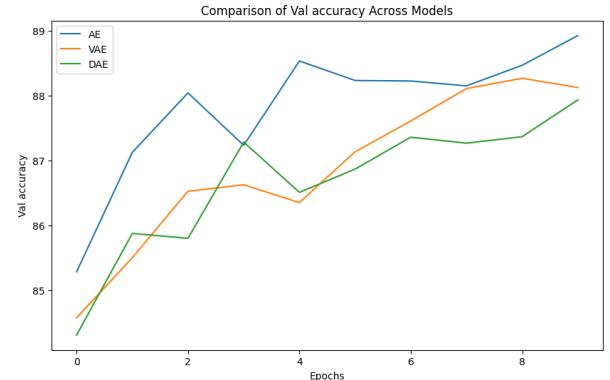
(a) Train Loss



(b) Validation Loss



(c) Train Accuracy



(d) Validation Accuracy

Figure 6: Comparaçao de Métricas de treino: Loss and Accuracy.

3 Part II: Detecção de Objetos

Para usar o modelo de deteção YOLOv5 no subconjunto do dataset KITTI fornecido foi necessário criar um ficheiro .yaml que vai ter as diretórias do nosso Dataset. Foi também necessário dividir o dataset em Treino e Teste (80/20). Na primeira parte em que treinamos o modelo a partir do zero é pedido para treinar o modelo só para a classe "car", para isso foi necessário duplicar o dataset mas desta vez apenas com as imagens correspondentes à classe "car". A arquitetura usada para este problema foi a YOLOv5s.

3.1 Metricas

Utilizámos as seguintes métricas para avaliar ambos os testes:

- IoU (Intersection over Union) - mede a sobreposição entre a caixa prevista e a caixa ground-truth. Ou seja,

esta métrica dá-nos a informação que se a caixa desenhada está correta ou não.

- Precision - mede a fração das previsões corretas sobre todas as previsões.
- Recall - mede a fração dos verdadeiros objetos que foram corretamente detetados pelo modelo. Indica se o modelo tem a capacidade de identificar todos os objetos relevantes no modelo.
- Average Precision(AP): isto é a área sobre a curva Precision-Recall, calculada a partir de um threshold específico, neste caso ≥ 0.5 , esta métrica avalia o trade off entre Precision e Recall.
- Mean Average Precision(mAP) - é a média dos valores de AP ao longo de múltiplos IoU thresholds, neste caso entre 0.5 e 0.95.

3.2 Comparação entre os dois modelos

Table 6: Comparação de Resultados: Transfer Learning vs. Training from Scratch

| Métricas | Transfer Learning | Training from Scratch | Observações |
|--------------|-------------------|-----------------------|--|
| Precision | 0.8 | 0.6 | Transfer learning reduz os falsos positivos. |
| Recall | 0.6 | 0.4 | Melhor deteção de Objetos. |
| mAP@0.5 | 0.55 | 0.5 | Convergência mais rápida para o transfer Learning. |
| mAP@0.5:0.95 | 0.3 | 0.25 | Melhor localização da sobreposição. |

No transfer learning são usados pesos pré treinados, tipicamente treinados em grandes datasets, de modo a especificar o modelo numa tarefa específica. No caso do treino se pesos pré treinados, permite que o modelo aprenda somente pelo dataset fornecido.

No geral, o transfer learning, mostrou um desempenho relativamente superior quando comparado com o treino a partir do zero. Pois neste caso, o modelo tem um conhecimento à priori das features dos objetos, permitindo que reduza a taxa de falsos positivos no ínicio, enquanto treinando a partir do zero o modelo mostra dificuldades iniciais, mas melhora gradualmente. O Recall é consistentemente maior no transfer learning o que indica a habilidade do modelo reconhecer mais objetos. Observamos ainda que os resultados das curvas de loss no treino, são significativamente mais positivos para o caso do transfer learning. A análise destas métricas afirmam que a deteção de objetos como o modelo com os pesos pré-treinados trás vantagens em relação ao treino dos modelos treinados de raiz.



Figure 7: Classificação com pesos pre treinados

A Anexos - Outros Gráficos e Arquiteturas dos Modelos

A.1 Arquiteturas dos modelos

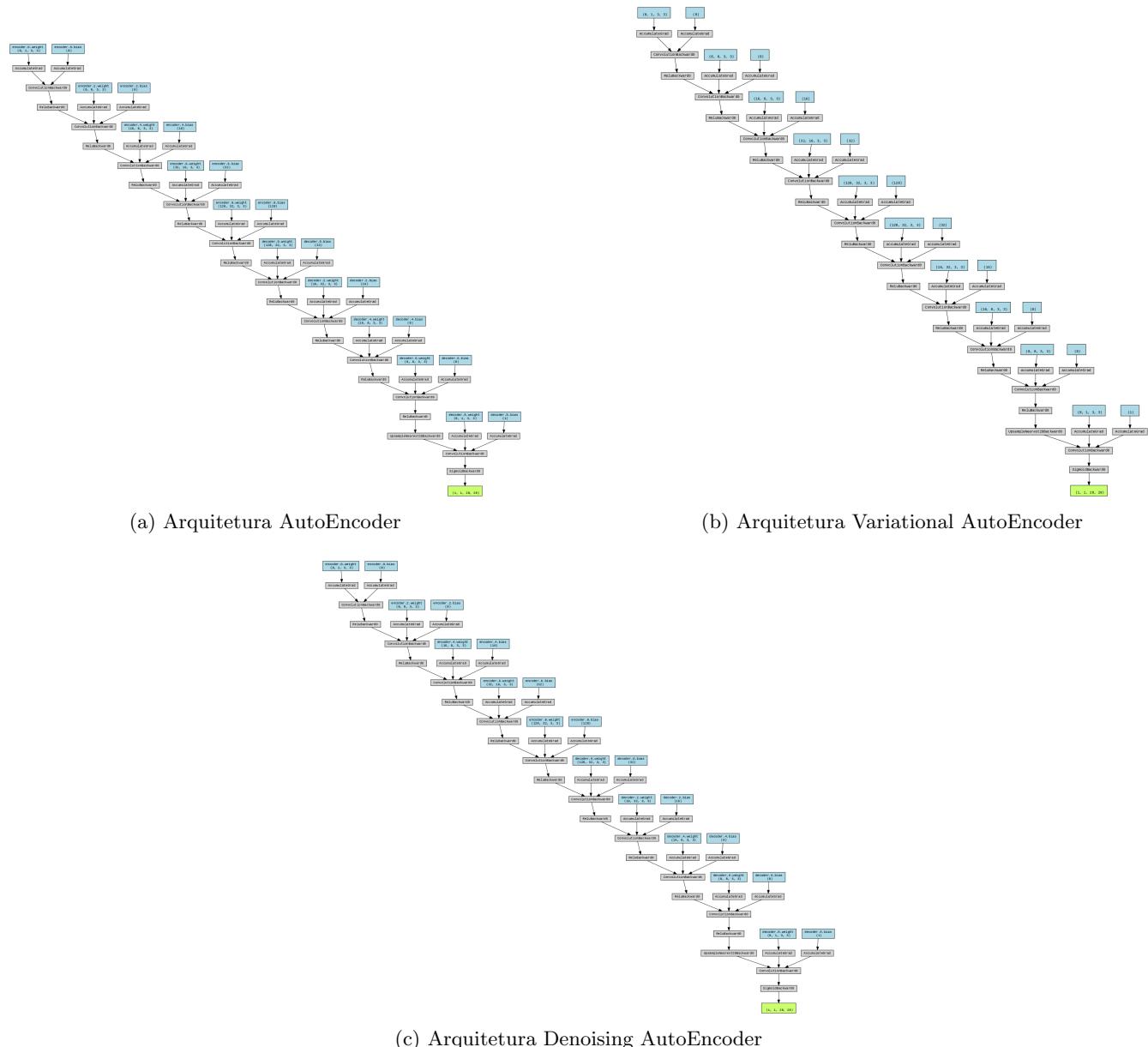
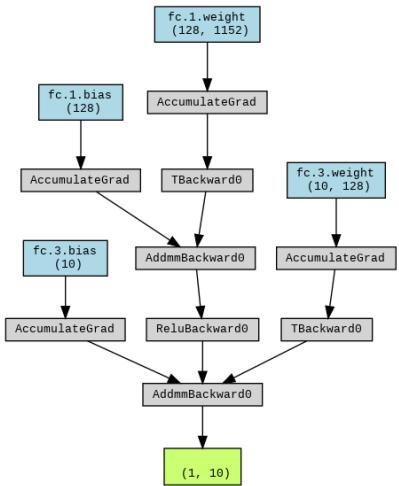
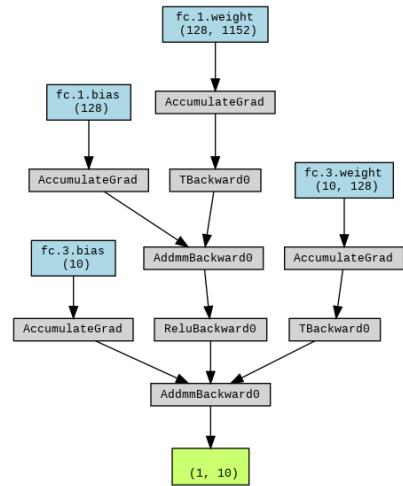


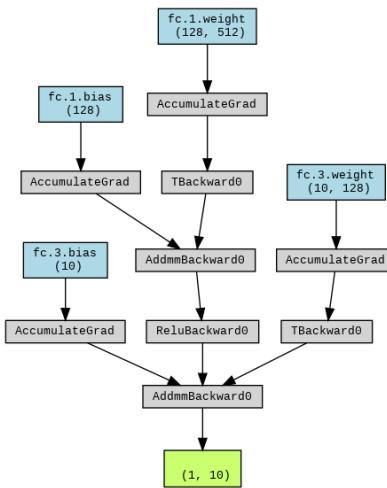
Figure 8: Comparação das Arquiteturas de AutoEncoder, Variational AutoEncoder, e Denoising AutoEncoder.



(a) Arquitetura AE Classifier



(b) Arquitetura DAE Classifier



(c) Arquitetura VAE Classifier

Figure 9: Comparaçao das Arquiteturas de Classificação para AE, VAE, e DAE.

A.2 Métricas YOLOv5 e Resultados

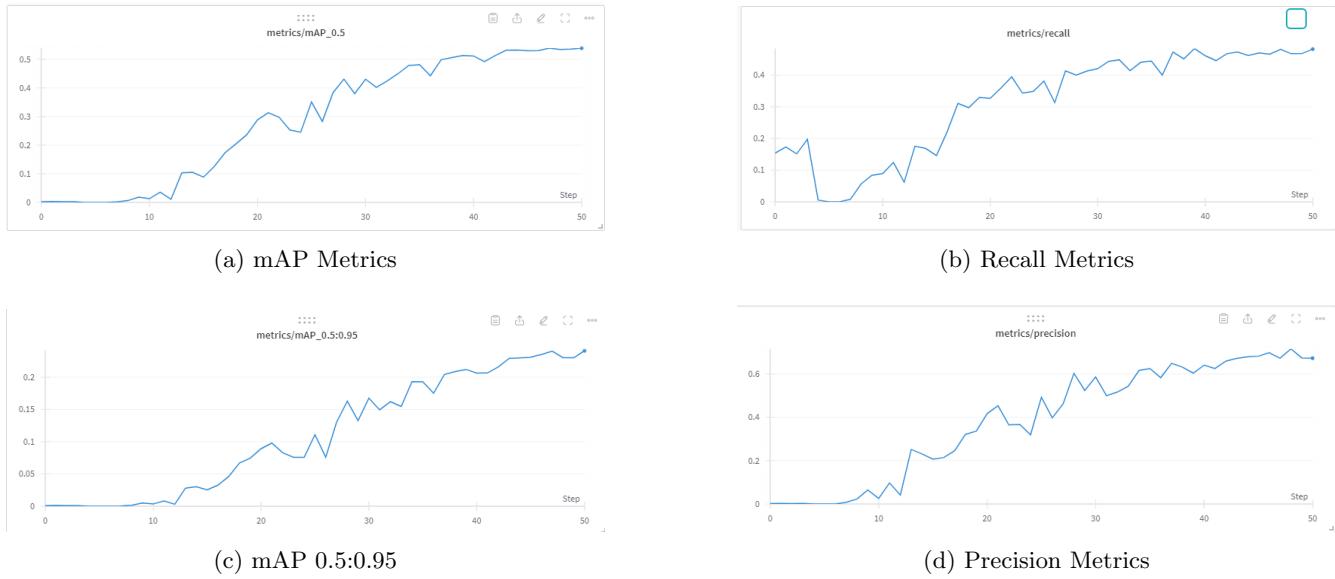


Figure 10: Comparação de Métricas - YOLOv5 From Scratch.

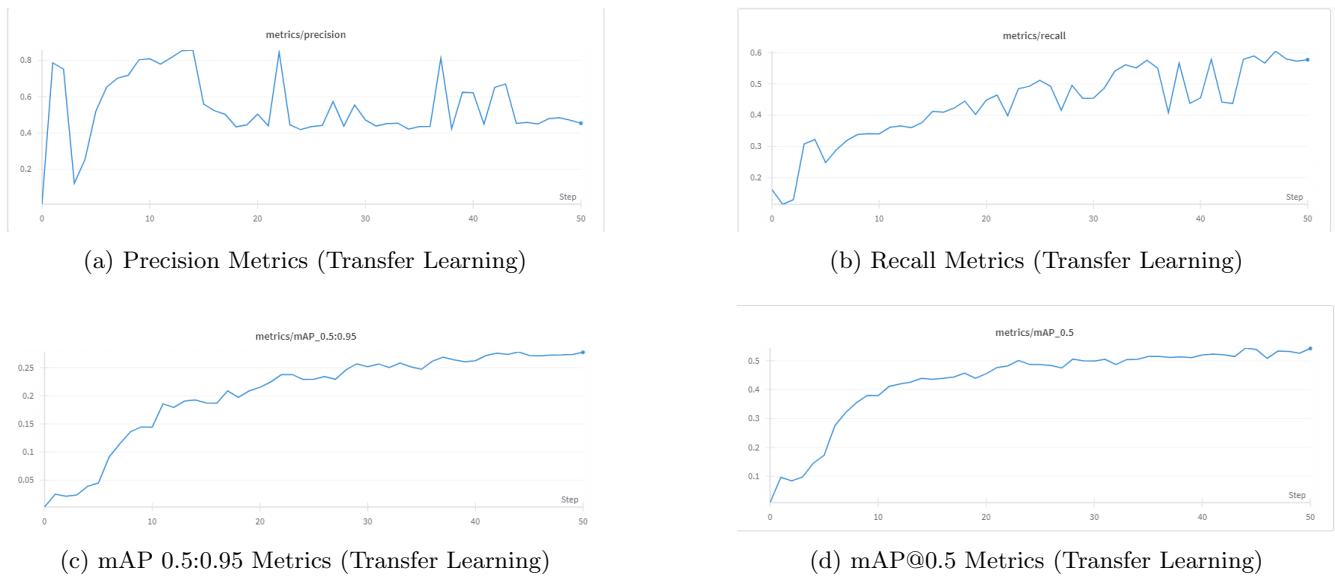


Figure 11: Comparação de Métricas: Precision, Recall, mAP@0.5:0.95, e mAP@0.5.



(a) Resultados da classificação treinado de Raiz



(b) Resultados Detecção com pesos pré-treinados

Figure 12: Comparação dos Resultados: Treinado de Raiz e com Pesos Pré-treinados.