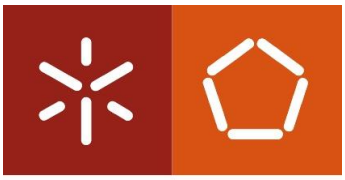

Universidade do Minho

Departamento de Informática

Tecnologias e Aplicações
Chinese MNIST Autoencoder

Jorge Mota (A85272)

July 22, 2021

Contents

Chapter 1

Introdução

Este projeto consiste no desenvolvimneto de redes generativas num contexto à escolha do aluno, e foi escolhido o desenvolvimento de autoencoders para caracteres chineses.

Autoencoders são tipos de redes neurais que aprendem a codificar de forma eficiente conjuntos de dados, usamos autoencoders concretos e variacionais, o primeiro extrai features discretas que constitui um espaço latente de vetores fixos ao invés do segundo, que compõe misturas de distribuições.

O dataset escolhido para este projeto foi o MNIST chinês, que incluí conjunto de figuras com números chineses escrito à mão.

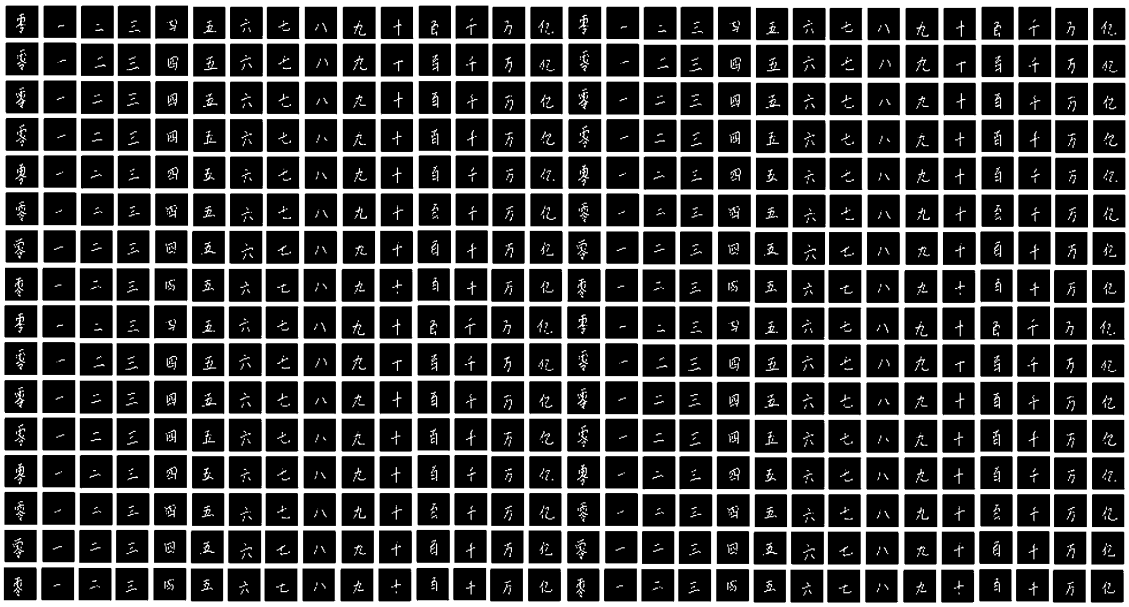

Figure 1.1: Pré-visualização do dataset

Chapter 2

Dataset

O principal caso de estudo é o dataset utilizado para treinar a rede neural, e para isto utilizei o *chinese MNIST dataset*, que contém cerca de 15'000 imagens de números escritos à mão. Existem 15 classes que descrevem númerações desde 0 até 10 e ainda 4 simbolos extra para numerar o 100, o 1'000, o 10'000 e o 100'000'000.

	value	character	code
0	0	零	1
1	1	一	2
2	2	二	3
3	3	三	4
4	4	四	5
5	5	五	6
6	6	六	7
7	7	七	8
8	8	八	9
9	9	九	10
10	10	十	11
11	100	百	12
12	1000	千	13
13	10000	万	14
14	100000000	亿	15

Figure 2.1: Legenda do dataset

Como podemos reparar imediatamente este dataset é bastante mais complexo que o MNIST tradicional, nao só por ter poucas mais classes, mas porque alguns simbolos têm carateristicas muito especificas e dificeis de codificar para todas as alternativas do dataset, olhemos por exemplo para o simbolo do 0:

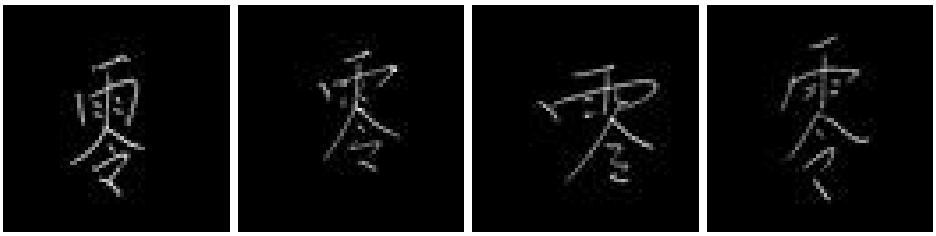

Figure 2.2: Variações de 0's

2.1 Preparação do dataset

Para preparar os dados para treino, compus um script (`prepare.py`) para dividir o dataset de forma justa para ter um *train set* e um *validation set* usado no treinamento com uma proporção de 70% e 30% respetivamente.

O produto do script de preparação separou o dataset na diretoria `chinese_mnist/` por `train_images` e `val_images`, em que cada uma destas diretorias tem pastas que identificam a label do conjunto de imagens que elas contêm (embora as labels não sejam utilizadas no treino da rede).

```
.
├── train_images
│   ├── 1
│   ├── 2
│   ...
│   ├── 14
│   └── 15
└── val_images
    ├── 1
    ├── 2
    ...
    ├── 14
    └── 15
```

Chapter 3

Concrete Autoencoder

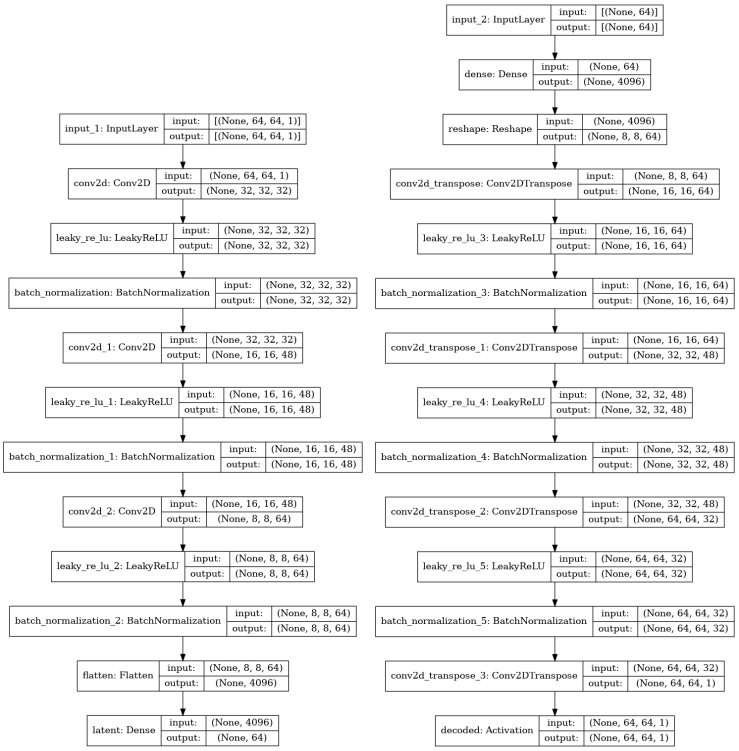

Figure 3.1: Autoencoder (Encoder e Decoder)

3.1 Arquitetura

O encoder desta rede é composto por três blocos convolucionais (`Conv2D` com função de ativação e `BatchNormalization`) de 64, 48 e 32 filtros. Em comparação com testes anteriores de redes para este dataset reduzi a quantidade de filtros convolucionais de 224 para 144, mantendo a fiabilidade dos resultados no mesmo patamar. A função de *loss* para esta rede é simplesmente a *loss* de reconstrução da imagem.

Como isto é um autoencoder concreto, a camada de código têm um espaço latente de tamanho 64 que dá um dominio grande o suficiente para captar os detalhes principais de cada tipo de caracter.

3.2 Treinamento

Para treinar esta rede, como é um tipo de aprendizagem não supervisionada, não há necessidade de passar a label ao processo de treinamento .

Foram realizados alguns treinos com cerca de 20 e 25 *epochs* e o *dataset* de treino for agrupado em 8 *batches*.

Como podemos notar pelas dimenções de cada imagem do dataset 64 de altura e comprimento é relativamente grande para ser processado, mas tive que manter esta componente desta maneira devido à perda de alguns detalhes importantes nmos caracteres quando se realiza um redimencionamento das imagens.

3.3 Resultados

Os resultados foram bastante satisfatórios, visto que não só a rede **reconstrói** a imagem codificada, como também elimina algum ruído da imagem original e **alisa** os traços mais significativos.

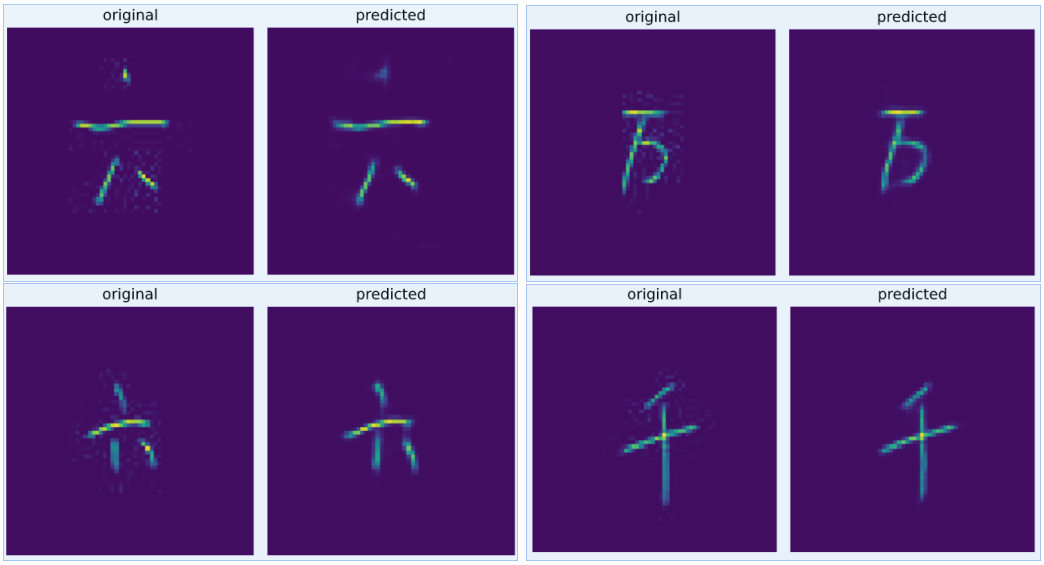

Figure 3.2: Original vs Reconstruida

Apesar de os resultados em geral reconstruírem bastante bem a imagem original, há ainda alguns simbolos (nomeadamente o '0') que expõem a limitação da rede de reconstruír de forma precisa todos os traços do caracter:

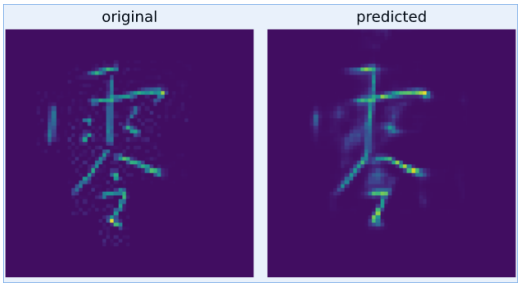

Figure 3.3: Original vs Reconstruida

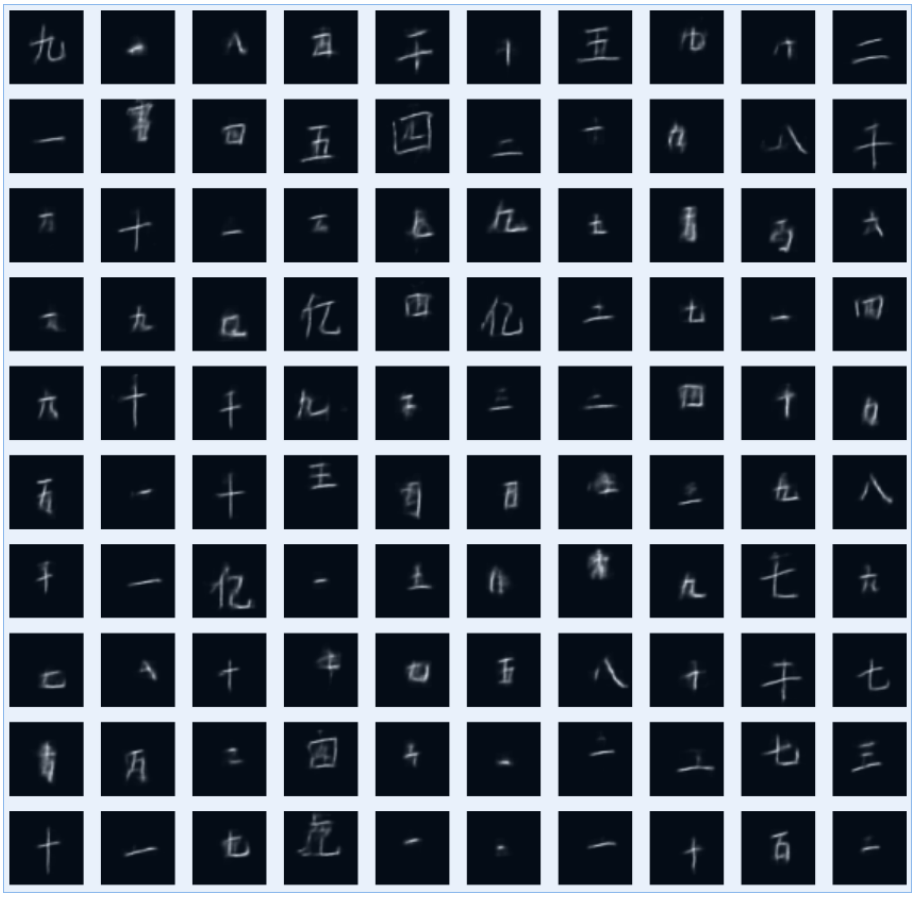

Figure 3.4: Amostragem de reconstruções do dataset

Chapter 4

Variational Autoencoder

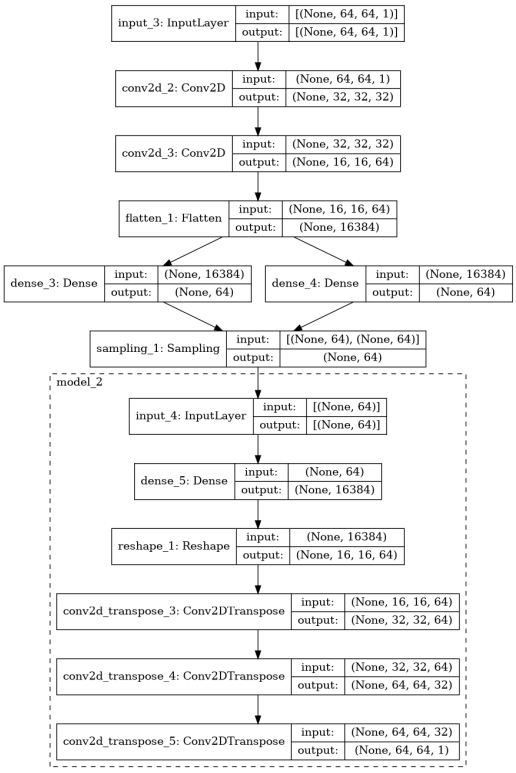

Figure 4.1: Autoencoder (Encoder e Decoder)

4.1 Arquitetura

Um autoencoder variacional, é uma rede semelhante à primeira, mas que em vez de codificar de forma concreta num vetor fixo o espaço latente, mapeia a imagem para um vetor de distribuições, o decoder vai receber uma amostragem dessas distruibuições como se fosse o vetor latente como na primeira rede.

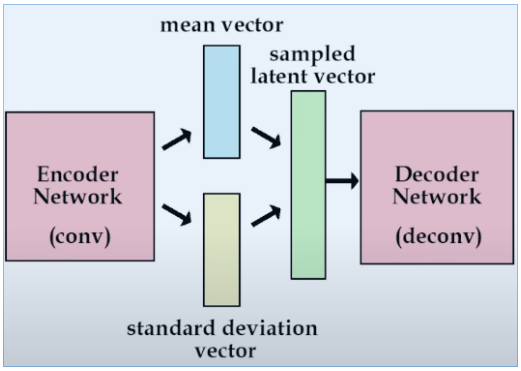

Figure 4.2: VAE latent space distributions

4.2 Treinamento

Foram realizados alguns treinos com cerca de 25 e 40 *epochs* e o *dataset* de treino for agrupado em 64 *batches*.

Para poder treinar esta rede, foi preciso especificar um procedimento novo manualmente devido à incapacidade de aplicar *backpropagation* à distribuição pela amostragem, por isso utilizou-se uma função *loss* que associa a *loss* de reconstrução e a divergência de KL.

4.3 Resultados

Os resultados foram aceitávelmente positivos, com o autoencoder a reconstruir os caracteres com alguma imprecisão, ainda assim nota-se que o autoencoder codifica boas propriedades e que as distribuições fazem o seu trabalho.

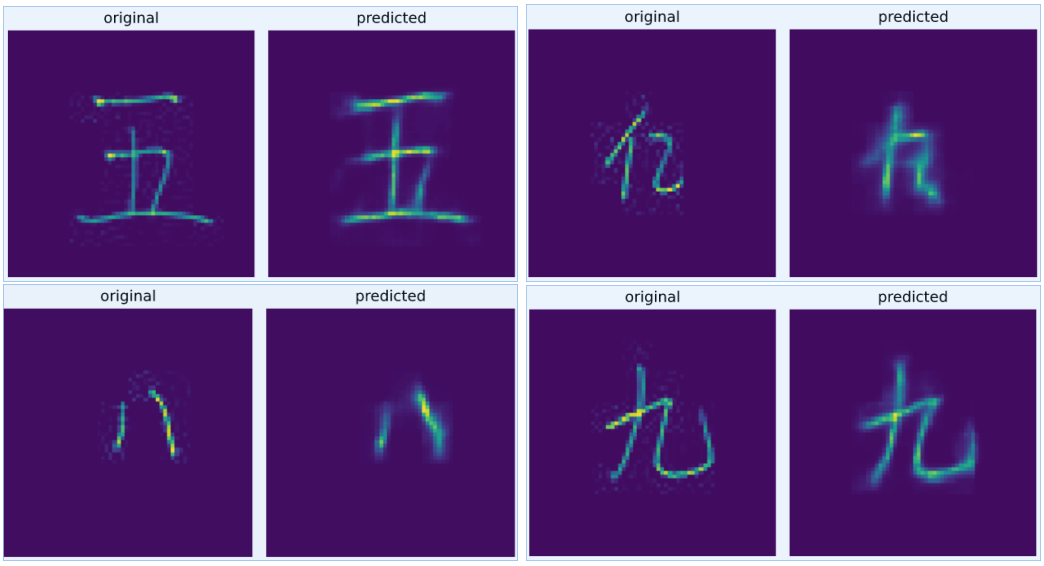

Figure 4.3: Original vs Reconstruida

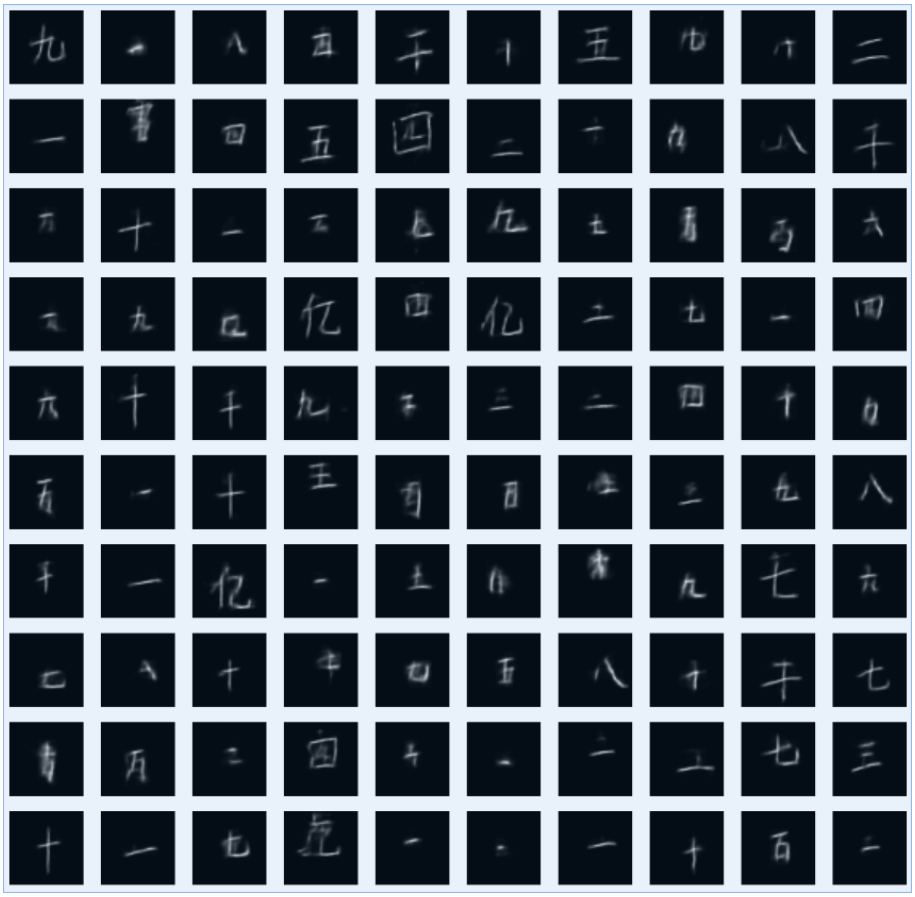

Figure 4.4: Amostragem de reconstruções do dataset

Chapter 5

Conclusão

O objetivo deste trabalho incluía explorar a natureza de modelos generativos, e para isso foi desenvolvido tanto um autoencoder concreto para a aproximação da função identidade na codificação das imagens do dataset utilizado, como também feito um autoencoder variacional que é um dos tipos de redes usadas hoje em dia para modelos generativos complexos.

Os objetivos do trabalho foram cumpridos com sucesso e as implementações funcionam de acordo com o que era esperado, para trabalho futuro o autoencoder variacional seria convertido para um autoencoder variacional condicional, para poder especificar a classe no decoder e obter as variantes para essa classe.