# Sistemas Autónomos - Apoio ao desenvolvimento de arquiteturas de redes CNN através da visualização de Featuremaps

**Diogo F. Braga**
University of Minho
Department of Informatics
4710-057 Braga, Portugal
Email: a82547@alunos.uminho.pt

***Resumo***. *As ações processadas pela visão humana têm sido transferidas para a área da Informática no sentido de potenciar a sua evolução. Este processamento é realizado informaticamente através da Visão Computacional, sendo o principal objetivo destes problemas usar os dados de qualquer imagem observada para inferir algo sobre o mundo. A evolução deste processo pode ser analisada através de várias técnicas de apoio ao desenvolvimento de redes, de forma a potenciar a sua eficiência.*

## 1 Introdução

A classificação de imagens é um processo de extração e interpretação de informação a partir de dados digitais. Este processo inclui-se na área científico-tecnológica da Visão Computacional, uma das principais sub-áreas da Inteligência Artificial. O principal objetivo deste processo é emular a visão humana e, assim, ter capacidade de interpretar imagens. Este fenómeno é normalmente realizado através da utilização de *Convolutional Neural Networks (CNN)* sendo que, neste documento, vai ser abordada uma técnica que suporta o desenvolvimento das redes: visualização de *featuremaps*.

A demonstração é realizada sobre dois diferentes *datasets*, um na escala do preto e branco (*Mnist Fashion*) e outro mais complexo a cores (*CINIC-10*). Para cada uma das fases são apresentados os *featuremaps* obtidos de forma a evidenciar o processo de treino realizado na arquitetura da rede em questão, assim como uma análise sobre o mesmo. Este documento apresenta alguns relacionamentos com o primeiro projeto "CNNs vs MLPs na classificação de imagens", pelo que é importante ter esses conceitos presentes.

## 2 Mnist Fashion

No projeto anterior foi selecionada como melhor arquitetura a apresentada na seguinte figura 1. Esta possui duas camadas de convolução e *pooling*, seguido de uma camada de *dropout* e da *multilayer perceptron*, para classificação da imagem.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 24, 24, 30) | 780 |
| max_pooling2d_1 (MaxPooling2 | (None, 12, 12, 30) | 0 |
| conv2d_2 (Conv2D) | (None, 10, 10, 15) | 4065 |
| max_pooling2d_2 (MaxPooling2 | (None, 5, 5, 15) | 0 |
| dropout_1 (Dropout) | (None, 5, 5, 15) | 0 |
| flatten_1 (Flatten) | (None, 375) | 0 |
| dense_1 (Dense) | (None, 128) | 48128 |
| dense_2 (Dense) | (None, 50) | 6450 |
| dense_3 (Dense) | (None, 10) | 510 |

Fig. 1. Arquitetura da melhor solução obtida para o Mnist Fashion no projeto anterior

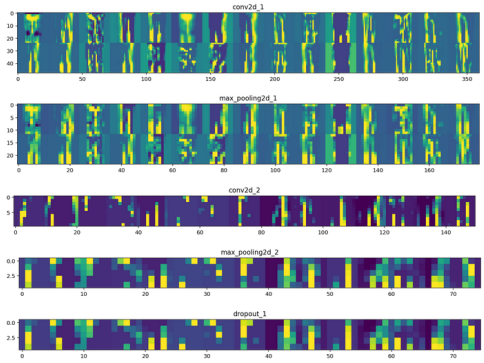


Fig. 2. Featuremaps da melhor solução obtida para o Mnist Fashion no projeto anterior, onde é possível visualizar a categoria 'calças'

Estas camadas criam e alteram os *featuremaps* durante o processo de treino. O efeito resultante de cada camada é demonstrado na figura 2. Na camada de convolução são

salientadas as *features* mais importantes das imagens. Na camada de *pooling* a imagem é reduzida, salientando apenas as *features* com maior relevância. Depois, este processo é repetido, mas para um número menor de *featuremaps*, com menos pixeis e canais de ativação, resultado dos processos anteriormente realizados. Em detalhe, o que se nota nestas fases é o refinamento das *features* constituintes das imagens.

Neste processo não existe qualquer *featuremap* a nulo e na última camada os *featuremaps* parecem ainda poder ser explorados, pelo que a arquitetura foi tornada um pouco mais complexa. A *accuracy* associada a este modelo foi de **90.14%**, mas com algum *overfitting* associado. A nova arquitetura contempla três séries, tendo cada uma: duas camadas de convolução, uma de *pooling* e outra de *dropout*. Deste modo explora-se mais o refinamento das *features*, realizando um processamento mais profundo. Esta arquitetura é apresentada na figura 3.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 28, 28, 32) | 320 |
| conv2d_2 (Conv2D) | (None, 26, 26, 32) | 9248 |
| max_pooling2d_1 (MaxPooling2 | (None, 13, 13, 32) | 0 |
| dropout_1 (Dropout) | (None, 13, 13, 32) | 0 |
| conv2d_3 (Conv2D) | (None, 13, 13, 64) | 18496 |
| conv2d_4 (Conv2D) | (None, 11, 11, 64) | 36928 |
| max_pooling2d_2 (MaxPooling2 | (None, 5, 5, 64) | 0 |
| dropout_2 (Dropout) | (None, 5, 5, 64) | 0 |
| conv2d_5 (Conv2D) | (None, 5, 5, 64) | 36928 |
| conv2d_6 (Conv2D) | (None, 3, 3, 64) | 36928 |
| max_pooling2d_3 (MaxPooling2 | (None, 1, 1, 64) | 0 |
| dropout_3 (Dropout) | (None, 1, 1, 64) | 0 |
| flatten_1 (Flatten) | (None, 64) | 0 |
| dense_1 (Dense) | (None, 512) | 33280 |
| dropout_4 (Dropout) | (None, 512) | 0 |
| dense_2 (Dense) | (None, 10) | 5130 |

Fig. 3. Arquitetura mais complexa para o Mnist Fashion

Os *featuremaps* correspondentes a esta arquitetura podem ser visualizados nas figuras 4 e 5. Na primeira camada de convolução é possível visualizar um *featuremap* nulo, significando isto que não foi ativado. Neste caso, a função de ativação é a *relu*, o que indica que esse *featuremap* era um valor negativo, e por isso não foi ativado. O refinamento de *features* continua até que na segunda série já se visualizam ainda mais *featuremaps* nulos. Este acontecimento é expectável pois, com o refinamento das *features*, os pormenores vão deixando de ter importância e são calculados como nulos. Na terceira série isto torna-se ainda mais evidente, sendo a maioria dos *featuremaps* nulos, com o *max pooling* a ser até totalmente nulo. Destas imagens conclui-se que a camada de convolução 6 já não deveria ser utilizada, devido à existência de muitos *featuremaps* a nulo. O mesmo acontece com as camadas seguintes e, apesar de na imagem só ser visível a camada 3 de *pooling*, a camada seguinte de *dropout* encontra-se igual pois esta nunca altera o resultado, devido a só ser utilizada no treino e não na avaliação nem na previsão. No entanto, antes de retiradas, a influência destas camadas finais deve ser verificada nos outros casos possíveis, pois para esses podem possuir um papel importante. A *accuracy* associada a este modelo foi de **90.88%** e, apesar de não ter muita diferença em relação ao anterior, constatou-se que o *overfitting* diminuiu, sendo por isso um ponto vantajoso e que faz esta arquitetura compensar.

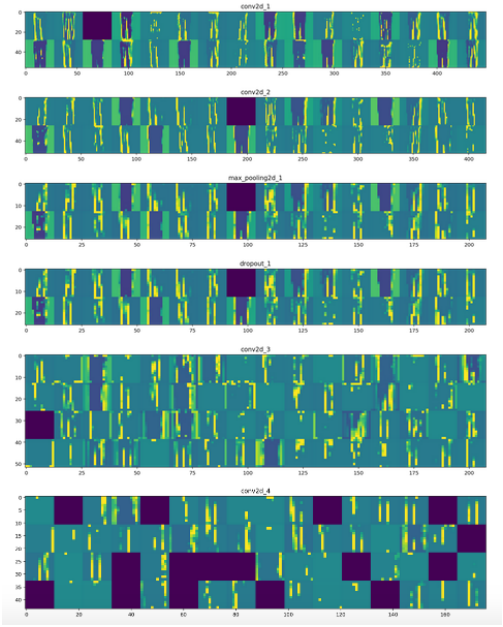


Fig. 4. Featuremaps da arquitetura complexa para o Mnist Fashion, na categoria 'calças' (parte 1)

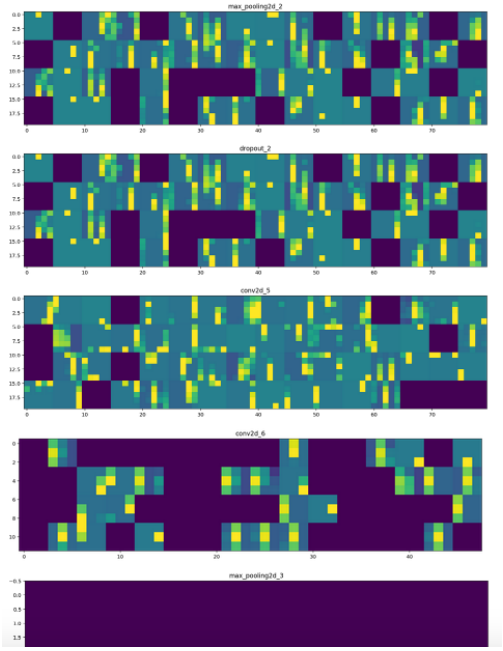


Fig. 5. Featuremaps da arquitetura complexa para o Mnist Fashion, na categoria 'calças' (parte 2)

## 3 CINIC-10

Para este *dataset* mais complexo, no projeto anterior foi selecionada como melhor arquitetura a apresentada na seguinte figura 6. Esta possui duas camadas de convolução com uma de *dropout* intermédia, seguido de uma camada de *pooling*, e da *multilayer perceptron*, para classificação da imagem. O efeito resultante de cada camada é demonstrado na figura 7.

Nas fases iniciais, ainda é possível visualizar algumas formas da figura do sapo, categoria que se encontra sob teste. Por outro lado, algumas possuem poucos detalhes em relação à imagem original. Nesta camada é possível visualizar alguns *featuremaps* com valor nulo, devido ao resultado com a função de ativação. Com o avanço do processamento, as ativações tornam-se mais abstratas e menos interpretáveis visualmente, pois começam a interpretar conceitos de nível mais baixo como curvas e ângulos, informação relevante para identificação da classe a que pertence. Os *featuremaps* nulos deixam de existir na terceira camada devido ao novo processo de convolução realizado que, com um novo filtro, percorre a matriz e dá origem a um novo mapa de ativação. A diminuição do tamanho de imagem também é uma consequência deste processamento. A accuracy associado a este modelo foi de **38.8%**, considerada bastante baixa. No entanto, importa fazer referência à execução de apenas 15 épocas de treino, que são consideradas curtas pois a *validation_loss* ainda se encontrava em curva descendente, e importa também fazer referência à grande complexidade das imagens deste *dataset*, que leva a concluir que uma rede mais complexa atingiria melhores resultados. Evoluiu-se nesse sentido, sendo a nova arquitetura apresentada na figura 8.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 32, 32, 32) | 896 |
| dropout_1 (Dropout) | (None, 32, 32, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 32, 32, 32) | 9248 |
| max_pooling2d_1 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| flatten_1 (Flatten) | (None, 8192) | 0 |
| dense_1 (Dense) | (None, 512) | 4194816 |
| dropout_2 (Dropout) | (None, 512) | 0 |
| dense_2 (Dense) | (None, 10) | 5130 |

Fig. 6. Arquitetura da melhor solução obtida para o CINIC-10 no projeto anterior

Os *featuremaps* correspondentes a esta nova arquitetura podem ser visualizados nas figuras 9 e 10. Uma arquitetura mais complexa tem impacto direto no refinamento das *features*, pois torna esta pesquisa mais profunda, na procura de detalhes que a visão humana não consegue identificar facilmente. Esta arquitetura contempla três séries, tendo cada uma: duas camadas de convolução, uma de *pooling* e outra de *dropout*. Importante referir que o otimizador da rede, que na arquitetura anterior era um *SGD* com taxa de aprendizagem de 0.0001, foi alterado para o *adam*, de forma a automatizar melhor este hiperparâmetro.

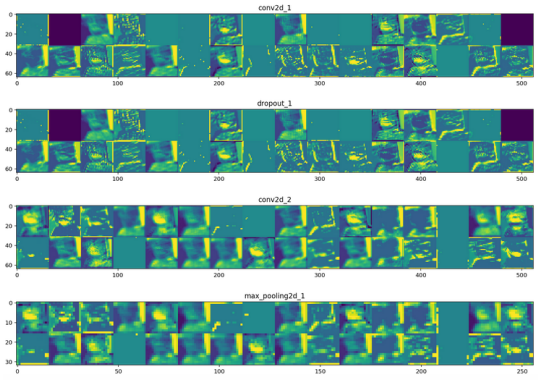


Fig. 7. Featuremaps da melhor solução obtida para o CINIC-10 no projeto anterior, onde é possível verificar a categoria 'sapos'

Nas camadas iniciais, em alguns dos *featuremaps* é possível visualizar bem as formas de um camião, sendo estas imagens ainda de alto nível para a rede. A partir da segunda série de camadas as imagens ficam menos reconhecíveis, devido aos processos de convolução e de *pooling*, que refinam as *features* dando origem a novos mapas de ativação com imagens mais pequenas. Devido à complexidade deste *dataset*, existem logo desde início bastantes *featuremaps* com valor nulo, resultado da função de ativação com valor negativo ou nulo. Este processo é semelhante aos apresentados anteriormente mas, ao contrário dos restantes, neste modelo notou-se uma evolução mais acentuada na *accuracy*, com valores por volta dos **51.8%** ao fim de 15 épocas. Dada a extrema complexidade do *dataset* e a continuidade descendente das curvas de *loss*, este processo deveria ser continuado no sentido de produzir melhores resultados na rede. Por motivos de *hardware* esta evolução tornou-se complicada, mas evoluiu-se um pouco mais a arquitetura de modo a verificar esses resultados.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_7 (Conv2D) | (None, 32, 32, 32) | 896 |
| conv2d_8 (Conv2D) | (None, 30, 30, 32) | 9248 |
| max_pooling2d_4 (MaxPooling2 | (None, 15, 15, 32) | 0 |
| dropout_5 (Dropout) | (None, 15, 15, 32) | 0 |
| conv2d_9 (Conv2D) | (None, 15, 15, 64) | 18496 |
| conv2d_10 (Conv2D) | (None, 13, 13, 64) | 36928 |
| max_pooling2d_5 (MaxPooling2 | (None, 6, 6, 64) | 0 |
| dropout_6 (Dropout) | (None, 6, 6, 64) | 0 |
| conv2d_11 (Conv2D) | (None, 6, 6, 64) | 36928 |
| conv2d_12 (Conv2D) | (None, 4, 4, 64) | 36928 |
| max_pooling2d_6 (MaxPooling2 | (None, 2, 2, 64) | 0 |
| dropout_7 (Dropout) | (None, 2, 2, 64) | 0 |
| flatten_2 (Flatten) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 512) | 131584 |
| dropout_8 (Dropout) | (None, 512) | 0 |
| dense_3 (Dense) | (None, 10) | 5130 |

Fig. 8. Arquitetura mais complexa para o CINIC-10

De qualquer das formas, o modelo transmite boas perspetivas pois, neste exemplo em causa, apesar de não ter classificado bem o camião, a previsão realizada foi um carro, o que não é totalmente errado pois estes objetos possuem características em comum, como por exemplo as rodas e a sua forma paralelepípeda.

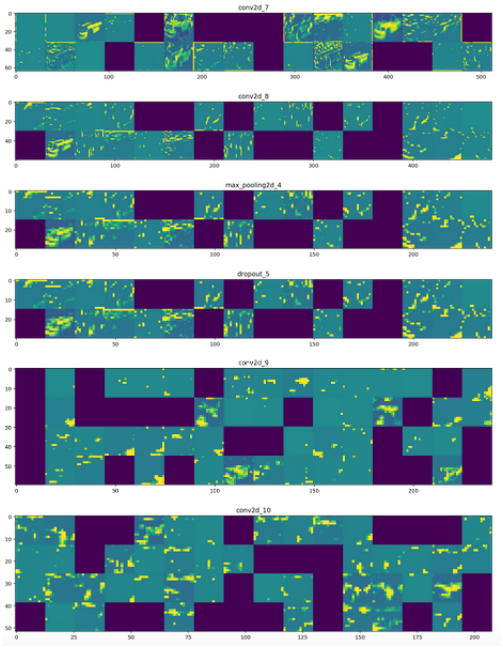


Fig. 9. Featuremaps da arquitetura complexa para o CINIC-10, na categoria 'camião' (parte 1)

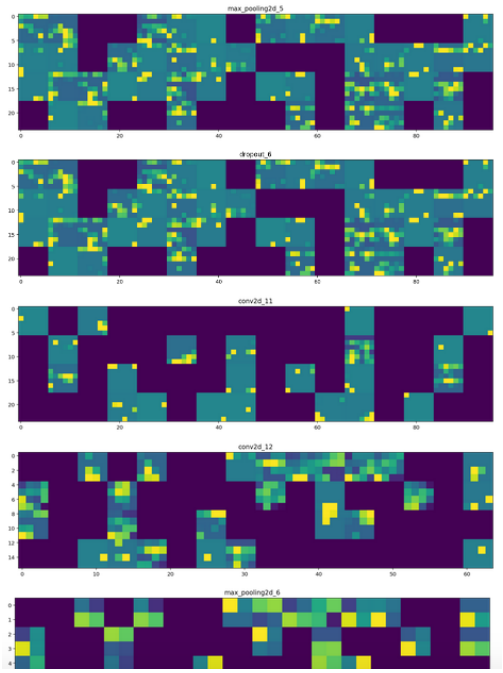


Fig. 10. Featuremaps da arquitetura complexa para o CINIC-10, na categoria 'camião' (parte 2)

A nova arquitetura é apresentada na figura 11. Esta rede, em relação à anterior, apresenta mais profundidade, possuindo 4 séries de camadas duplas de convolução, com camadas de *pooling* e *batch normalization*, estas últimas apenas introduzidas nesta arquitetura final. São também introduzidas algumas camadas de *dropout* para tentar diminuir o *overfitting*. Os *featuremaps* correspondentes a esta nova arquitetura podem ser visualizados nas figuras 12, 13 e 14.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 32, 32, 32) | 2432 |
| conv2d_2 (Conv2D) | (None, 32, 32, 32) | 25632 |
| max_pooling2d_1 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| batch_normalization_1 (Batch | (None, 16, 16, 32) | 128 |
| conv2d_3 (Conv2D) | (None, 16, 16, 64) | 18496 |
| conv2d_4 (Conv2D) | (None, 16, 16, 64) | 36928 |
| max_pooling2d_2 (MaxPooling2 | (None, 8, 8, 64) | 0 |
| batch_normalization_2 (Batch | (None, 8, 8, 64) | 256 |
| dropout_1 (Dropout) | (None, 8, 8, 64) | 0 |
| conv2d_5 (Conv2D) | (None, 8, 8, 64) | 36928 |
| conv2d_6 (Conv2D) | (None, 8, 8, 64) | 36928 |
| max_pooling2d_3 (MaxPooling2 | (None, 4, 4, 64) | 0 |
| batch_normalization_3 (Batch | (None, 4, 4, 64) | 256 |
| dropout_2 (Dropout) | (None, 4, 4, 64) | 0 |
| conv2d_7 (Conv2D) | (None, 4, 4, 128) | 32896 |
| dropout_3 (Dropout) | (None, 4, 4, 128) | 0 |
| conv2d_8 (Conv2D) | (None, 4, 4, 128) | 65664 |
| max_pooling2d_4 (MaxPooling2 | (None, 2, 2, 128) | 0 |
| batch_normalization_4 (Batch | (None, 2, 2, 128) | 512 |
| flatten_1 (Flatten) | (None, 512) | 0 |
| dropout_4 (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 1024) | 525312 |
| dropout_5 (Dropout) | (None, 1024) | 0 |
| dense_2 (Dense) | (None, 512) | 524800 |
| dense_3 (Dense) | (None, 10) | 5130 |

Fig. 11. Arquitetura final para o CINIC-10

Numa fase inicial ainda existem alguns *featuremaps* não ativados, sinal que não são ativações consideradas pela rede. Nestas camadas iniciais ainda é possível visualizar a forma do carro. Como a rede é mais profunda e refina as *features* de uma forma mais demorada, as restantes camadas apresentam na mesma bastante informação nos seus *featuremaps*, neste caso a um nível mais detalhado como ângulos e curvas, existindo poucos *featuremaps* sem ativação. Ao fim de 15 épocas, esta arquitetura apresentou uma *accuracy* de **69.1%**, mostrando-se assim vantajosa a sua utilização. Nesta arquitetura foram também introduzidas camadas *batch normalization*, pelo que a melhoria dos resultados podem também estar relacionados com esta utilização.

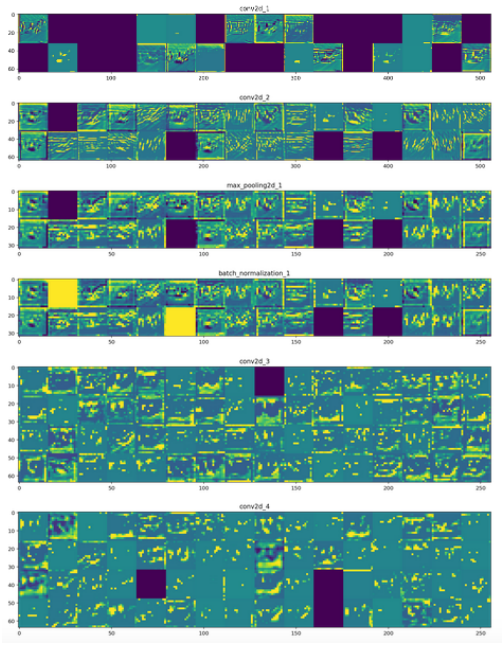


Fig. 12. Featuremaps da arquitetura final para o CINIC-10, na categoria 'carro' (parte 1)

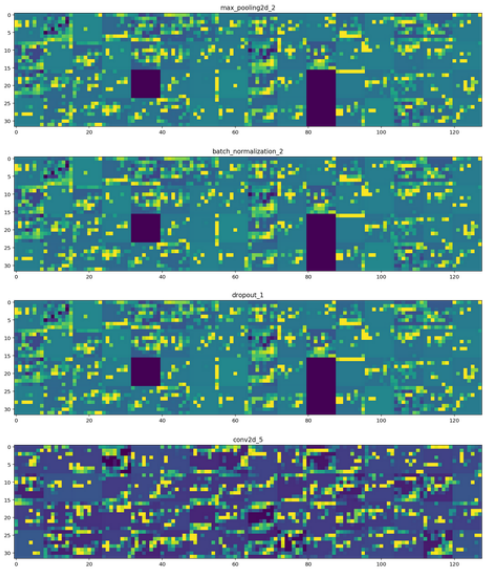


Fig. 13. Featuremaps da arquitetura final para o CINIC-10, na categoria 'carro' (parte 2)

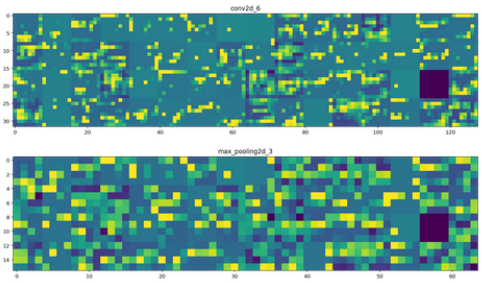


Fig. 14. Featuremaps da arquitetura final para o CINIC-10, na categoria 'carro' (parte 3)

## 4 Conclusão

Deste documento e pesquisas relacionadas é importante, tal como no projeto anterior, constatar a importância que o *hardware* possui na execução de CNNs, devido à grande quantidade de dados treinados e testados nas arquiteturas normalmente complexas neste tipo de rede. Este ponto foi uma limitação na realização de testes pois devido à falta de um GPU não foi possível aplicar outras bibliotecas com melhores rendimentos e, consequentemente, não foi possível gerar muitos testes para comparação nem conseguir visualizar o processo de treino com um bom ritmo.

Neste projeto a evolução das redes é principalmente realizada através do aumentar da profundidade da rede, de forma a realizar um maior refinamento das *features*. Posto isto, importante referir que ambas as arquiteturas podem ser evoluídas de outras formas no sentido de produzir melhores resultados pois, por exemplo, nota-se alguma frequência de *featuremaps* que não são ativados. Neste sentido, uma possível alteração seria aumentar o tamanho de *input* das imagens, que nestes testes foi de 32x32 pixeis, mas que se espera que, com uma resolução superior, proporcionasse às redes mais definição nas *features* das imagens e, consequentemente, estas possuíssem melhores resultados.

Para desenvolvimento das arquiteturas de redes CNN, a visualização de *featuremaps* torna-se uma técnicas muito vantajosa pois torna possível a compreensão do que acontece no processo de treino, especificamente em cada camada. O efeito destas e a utilidade que possuem para os dados em questão são capacidades que a visualização dos *featuremaps* proporcionam a quem tenta encontrar o melhor modelo possível para a classificação de imagens contidas num determinado *dataset*. É possível, de forma detalhada, visualizar o efeitos das camadas de convolução, de *pooling* e de *batch normalization*, estas que se apresentam como parte essencial na concenção de qualquer *Convolutional Neural Network*.