Overfitting to Success: A Pragmatic Approach to Increasing Categorical Accuracy in Simpsons Character Recognition

Bevan Fairleigh[*]
bfairleigh@deakin.edu.au
Deakin University
Burwood, Victoria, Australia

Until now, this was the only way to recognise Simpsons characters
You mean there's a better way?

Figure 1: A hilarious Simpsons *meme* created by the author during many hours of training image collection [

ABSTRACT

This report summarises the journey taken by the author to create and deploy an image classification model as part of the entry into Deakin University's Simpsons Challenge 2021. The report details steps taken to clean and supplement training data, the training process and model selection, and steps taken to improve the final model. The process generated several models with an equivalent categorical accuracy score on the hidden test data set of 99.3%, with one model scoring **99.1%** on the hidden final test data set. Some of the techniques used go against the traditional machine learning rules, which make for interesting reading.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; *Supervised learning by classification.*

[*]Individual effort

KEYWORDS

Supervised Learning, Computer Vision

ACM Reference Format:
Bevan Fairleigh. 2021. Overfitting to Success: A Pragmatic Approach to Increasing Categorical Accuracy in Simpsons Character Recognition. In *Deakin Simpsons Challenge, June, 2021, Melbourne, VIC.* , 5 pages.

1 INTRODUCTION

There are plenty of articles written about the *dos* and *don'ts* for Machine Learning. In my approach to the Deakin Simpsons Challenge 2021, I experimented with both to build a model which scored above 99% accuracy on all competition test sets. Admittedly, the most significant model improvements arrived through classic data preparation (cleansing, supplementation, and augmentation); followed by intuitive model selection and hyper-parameter tuning. I have thoroughly detailed all these steps within this report; however, as validated by a successful increase in accuracy on hidden data, sometimes, *"you have to break the rules to free your heart"* (Season 11/Episode 5).

A key machine learning fundamental I discerningly ignored was separating the training and validation data sets. In other words, I used the same data to train and to validate my model. In theory, this could lead to a scenario where a model becomes *overfit*; with the model *memorising* more than *learning*. Yet it improved my model's performance. In this report, I justify why in this scenario the decision made sense, and objectively delivered superior results.

Table 1: Xception Final Hyper-parameters

Hyper-parameter	Value	Comments
Learning Rate α	{1e-4,1e-5,1e-6}	Adjusted to find minimum
Input Shape	(224,224,3)	*Default for MobileNet*

Table 2: Training Data Augmentation

Augmentation	Value
Zoom range	0.6 - 1.8
Rotation range	35
Brightness range	0.6 - 1.8
Shear range	50
Width shift range	0.2
Height shift range	0.2
Horizontal flip	True

Before we begin, remember... This model achieved over 99% on the two competition test sets. Which is pretty good.

2 MODEL SUMMARY

For those non-Machine Learnists, this section contains technical information related to the final model. Section 3 is where the fun begins, detailing the process used to achieve +99%.

The final submission was built using the Keras Xception [Chollet 2016] architecture, initialised with pre-trained weights from ImageNet. Through transfer learning, keeping all layers unfrozen, the Xception model was trained on a heavily augmented training set.

2.1 Final Model Hyper-parameters

Hyper-parameters were tuned to find a minimal Validation Loss score, with final values shown in Table 1. I used RMSprop as the optimizer, gradually lowering the learning rate. In theory, RMSprop uses an adaptive learning rate, so this process was not guaranteed to yield superior results - but it did, and this is a competition (>99%).

In my experiments, activation by both standard ReLU and Leaky ReLU delivered statistically indistinguishable results. Ultimately the model submitted contained standard ReLU.

Image size was also increased to (224,224).

2.2 Final Data Augmentation

As indicated, my model was trained and validated on a complete non-split data set; however, the training data was augmented through Keras' Data Augmentation module. Importantly, the validation set was **not augmented**. Final perturbations are displayed in Table 2.

Example output of the data augmentation process is contained within Figure 2.

2.3 Final Model Results

Table 3 shows the results of the final model while training, and against the hidden test set contained on the Codalab competition platform. Figure 3 and Figure 4 show post training model results on Validation data.

Table 3: Final Model Results

Criterion	Score	Comments
Categorical Accuracy	98.39%	Training
Loss	0.0592	Training
Categorical Accuracy	99.99%	Validation
Loss	0.00036	Validation
Accuracy	99.3%	Test Open
Accuracy	99.1%	Test Final

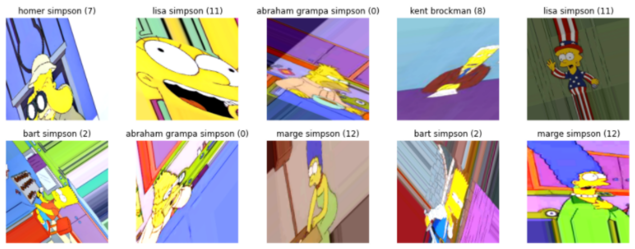


Figure 2: Example output of Data Augmentation process

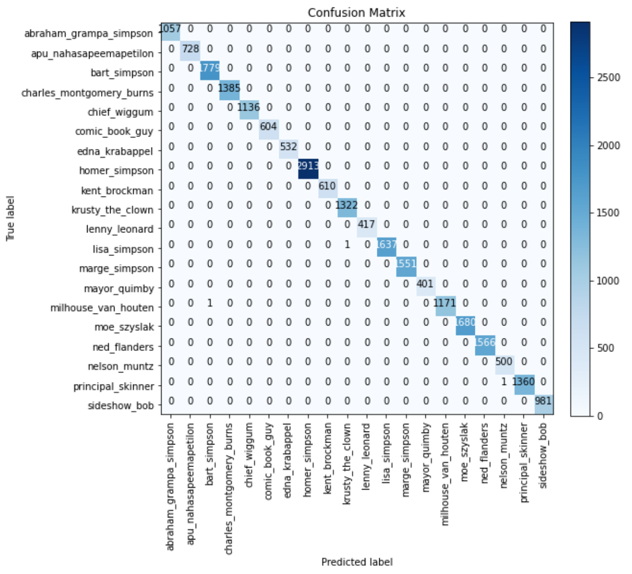
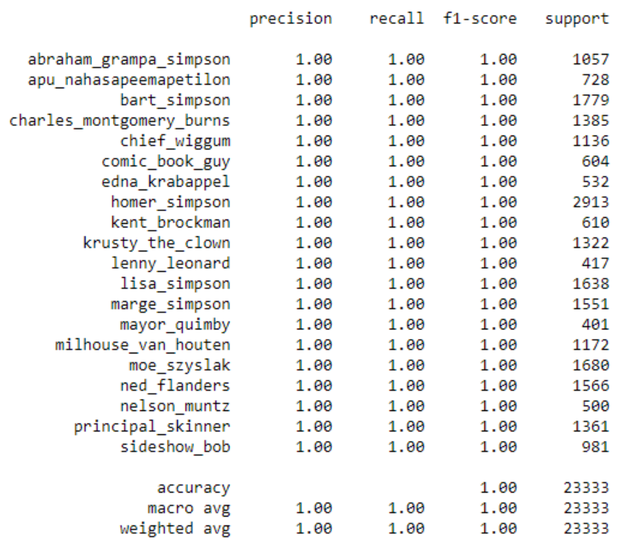


Figure 3: Confusion Matrix

3 PROCESS

This is the fun stuff.

The most significant improvements to the accuracy of my model came from textbook processes. I spent a significant amount of time performing data preparation including data cleansing, data supplementation and model weakness rectification, ensuring that the most appropriate model architecture was used, and finally, that my training process was successfully building towards an accurate model. While this section is presented in a linear fashion, I performed all steps concurrently and iteratively. As such, there



```
                          precision    recall  f1-score   support

   abraham_grampa_simpson       1.00      1.00      1.00      1057
   apu_nahasapeemapetilon       1.00      1.00      1.00       728
             bart_simpson       1.00      1.00      1.00      1779
charles_montgomery_burns       1.00      1.00      1.00      1385
             chief_wiggum       1.00      1.00      1.00      1136
           comic_book_guy       1.00      1.00      1.00       604
           edna_krabappel       1.00      1.00      1.00       532
            homer_simpson       1.00      1.00      1.00      2913
            kent_brockman       1.00      1.00      1.00       610
         krusty_the_clown       1.00      1.00      1.00      1322
            lenny_leonard       1.00      1.00      1.00       417
             lisa_simpson       1.00      1.00      1.00      1638
            marge_simpson       1.00      1.00      1.00      1551
             mayor_quimby       1.00      1.00      1.00       401
      milhouse_van_houten       1.00      1.00      1.00      1172
              moe_szyslak       1.00      1.00      1.00      1680
             ned_flanders       1.00      1.00      1.00      1566
             nelson_muntz       1.00      1.00      1.00       500
        principal_skinner       1.00      1.00      1.00      1361
             sideshow_bob       1.00      1.00      1.00       981

                 accuracy                           1.00     23333
                macro avg       1.00      1.00      1.00     23333
             weighted avg       1.00      1.00      1.00     23333
```

Figure 4: Classification Report

was no *silver bullet* that provided the most significant performance increase, instead, my model consistently improved over time until the competition final submissions were due.

To begin, I developed a very simple six layer feed-forward CNN model based on the background reading provided by the competition [Attia 2017], trained on the provided training data set. This delivered a model with just below 80% accuracy on competition data. It was a good starting point.

3.1 Data Preparation

3.1.1 Data Cleansing. The confusion table from the starting model showed plenty of confusion, especially between Lisa and Bart, and, Grampa Simpson and Homer. This led me to explore the data manually to discover there were numerous erroneous or ambiguous entries in the training data. To address this efficiently, I created a script (creatively called test.py) which could load a model to conduct classification on the training data set. Test.py would iterate through the input directory and relocate each image to the appropriate output directory (i.e. Homers into the homer_simpson directory). My starting model wasn't perfect (~80%) but it was accurate enough to confidently isolate images that were polluting my data set. These were rectified before retraining and improving my model.

3.1.2 Data Supplementation. In addition to cleansing, I identified many critical gaps in the training data. Homer made for a significant proportion of the training data size, with over 2000 images, while some minor characters had a much smaller representation. I started by supplementing these poorly represented characters by searching YouTube videos for "Best of *character X...*". After capturing the video and extracting frames, I had a large number of new images. Using my ever-improving test.py (and a manual check) to classify the newly collected images, I added them to my training data set. However, I needed more data.

After dusting off old Simpsons DVDs, I targeted episodes focusing on these under-represented characters (i.e. Sideshow Bob-centric episodes). This process boosted the characters of choice, while also supplementing the training data for other characters through incidental collect.

3.1.3 Model Weakness Rectification. I began to notice that many new images of characters were very similar to images already included in the training set. If the training data contained too many similar images, I would risk creating a model with unwelcome bias towards those similar images. This was exemplified by the misclassification of characters inside the Kwik-E-Mart. My model assumed most images inside the Kwik-E-Mart must be Apu, because most of Apu's training data was from inside the Kwik-E-Mart. To fix this, I needed to add more Apus from outside the Kwik-E-Mart, and more non-Apus from inside the Kwik-E-Mart.

Other examples were:

- Characters wearing orange shirts were classified as Bart
- Characters shown inside a TV Frame were classified as Kent Brockman
- Characters wearing hats were classified as Chief Wiggum
- Characters shown with Smithers were classified as Mr Burns

I prioritised finding images that avoided those learned character stereotypes. However, there was more that could be done to improve the variance within each character's training set. Ingeniously, I modified test.py to prefix the confidence score of each classification to the image filename. This resulted in one of four outcomes, all of which added significant value to my data supplementation process.

Each new image was either:

- Misclassed with a high confidence score - This meant my model was outright wrong. There is definitely something it could learn from this image.
- Misclassed with a low confidence score - This meant my model was wrong, but it probably knew it was wrong, and was being shy. There is something that it could learn from this image too.
- Correctly classed with a low confidence score - While correct, more confidence would be good, so again, my model could learn something from this image.
- Correctly classed with a high confidence score - Congratulations model, you've done well and you have nothing to learn from this image.

In the first three cases, the newly classified images were added into the training set. As my data set grew towards 23,333 images, the bias that was negatively impacting my accuracy results began to subside.

3.2 Training Process

3.2.1 Overfitting. As confessed in the introduction, I took deliberate action to combine the training and validation data sets (although with data augmentation applied to the training set). I made this decision because some images containing higher variance (i.e. unique images) were being randomly excluded from the training set during the split. I wanted to find a way to ensure all of my valuable training data collected during the model weakness rectification process (section 4.3) were included. I knew that by using the same data set to train and validate could lead to overfitting; however, I hypothesised three reasons to explain why this process would improve my model:

- The set of Simpsons characters is finite. Unlike photographic images of the natural world, there is only a limited set (albeit, a very large set) of images of Simpsons characters. As a training set approaches the total number of possible Simpsons images, an overfit model's accuracy **must** increase, as the likelihood of training images existing within the competition test set would approach 100%.
- Each Simpsons character is defined by a strict set of design guidelines [Weinstein 2020]. These design rules indicate that variability between images is generally low. Even with a degree of overfitting, good generalisation can still be achieved because variability within each character set is low.
- Except when variability is high! In some episodes, characters are modified away from their stereotype in dramatic fashion. e.g. There have been 31 episodes of 'Treehouse of Horror' where characters often appear with significant variance. I needed to ensure these high variability events were captured.

My hypothesis was correct, my model generated a higher validation accuracy score and competition test score (>99%). When using this model with test.ps to collect further data, both accuracy and confidence scores were much higher on all truly unseen data. Although unconventional, objectively it was the right choice as it squeezed the absolute maximal result from my model and I moved up the leaderboard.

3.2.2 *Data Augmentation.*

While the training and validation sets were initially identical, I ensured the training set was significantly modified through data augmentation. Data augmentation, as I had previously incorrectly assumed, does not *augment* the set, instead it replaces it with the modified images [Brownlee 2019]. This gave me confidence that my process as detailed in 3.2.1 would not lead to significant overfitting, as both training and validation sets were now different.

Initially, I selected the values for augmentation by what I would reasonably expect to see during a Simpsons episode. I continued to increase the perturbations to find the maximum delta from the original image, while still producing a model that could accurately classify non-perturbed images. Interestingly, as I continued tweaking these parameters, I discovered my model performed better at generalisation than on the training data. This highly favourable symptom was present even without the overfitting process from section 3.2.1. Training history is shown in figure 5.

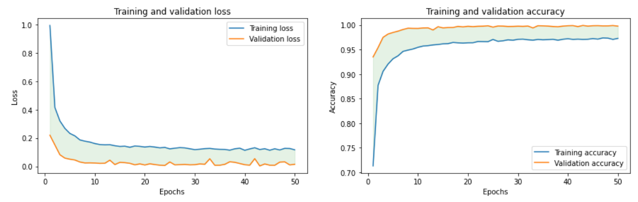


Figure 5: History of training, showing Validation results better than training results

3.2.3 *Model Selection.*

During my testing, I experimented with multiple models, including MobileNet, VGG16 and RESNET152v2. These models were discarded primarily because they were not able to achieve the level of accuracy I found with the Xception model, or training time was significantly longer.

I also tested an Xception model built layer-by-layer and trained scratch, but it did not compare to the speed of training the pre-trained Imagenet Xception model. Using this pre-trained model, I was able to achieve a validation accuracy of 93% after only one epoch, which significantly enhanced the training time while tuning other parameters. I tried improving training speed further by freezing several layers of the pre-trained model, but this did not perform well in accuracy. This is likely because Simpsons characters are intrinsically different to photographic images as contained in the Imagenet data set, so while the pre-trained weights are a good start, they are not ultimately suited to cartoons. As shown in section 3.2.2 my model generalised very well, so there was no need for any additional dropout layers, and when trialled, they did not improve performance.

While I was testing the MobileNet pre-trained model, I experimented with the default input size of 224 x 224. Increasing the number of inputs increased my model's training time, but gave my model more data points on which to make decisions. I tested this increased size on the Xception model, and noticed a significant improvement in model accuracy, so I selected the input size of (224,224,3).

3.2.4 *Training Runs.*

With all parameters finalised and my training set ready, I launched a final training run of 50 epochs, saving the model which achieved the lowest Validation Loss score. Upon completion, I lowered the learning rate and re-ran training for another 10 epochs, before lowering the learning rate and re-running one final time.

3.2.5 *Final Submission.*

The processes outlined in this report delivered a number of different but high performing models. Several of my models scored 99.2% on the competition test data, while two models scored 99.3%. Upon entry into the final round, my two most precious models scored 99.0% and 99.1% respectively. Which once again, was outstandingly excellent.

4 CONCLUSION

This report documents my efforts to create an extremely accurate and easily repeatable Simpsons character classification model. As evidenced, every decision I made on the journey was a considered one, ensuring that my model's behaviour was always explainable. While most of these decisions were textbook examples of *dos*, I most definitely experimented with the *don'ts*, and my results are stronger because of it. After all, this was a competition, and my methodical approach to this experimentation resulted in an observable increase in accuracy, scoring greater than 99% against all competition test sets.

ACKNOWLEDGMENTS

Thank you to:

Deakin University School of Information Technology, Faculty of Science, Engineering & Built Environment (SEBE), Competition facilitators Dr. Mohamed Reda Bouadjenek, Thuy Nguyen, Prof. Peter Eklund and Dr. Sunil Aryal

Competition sponsor Bendigo Bank Community Bank Deakin University, and,

The other competitors in the competition, who all displayed exceptional competitiveness and camaraderie throughout this adventure.

REFERENCES

Alexandre Attia. 2017. *The Simpsons characters recognition and detection using Keras.* Retrieved March 10, 2021 from https://medium.com/alex-attia-blog/the-simpsons-character-recognition-using-keras-d8e1796eae36/

Dr Jason Brownlee. 2019. *How to Configure Image Data Augmentation in Keras.* Retrieved April 30, 2021 from https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

François Chollet. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR* abs/1610.02357 (2016). arXiv:1610.02357 http://arxiv.org/abs/1610.02357

Josh Weinstein. 2020. *What you will never see on the Simpsons (from a delightful, nearly 500 page early style guide, circa 1990).* Retrieved May 24, 2021 from https://twitter.com/Joshstrangehill/status/1252750803586002944

A SOURCE CODE

Bevan Fairleigh's Github : https://github.com/bevanyeah/deakin-simpsons-challenge2021

Training Data : https://drive.google.com/file/d/1mbdOMc6bh4HlKbH0TgSuJ2CXSM47dZbL/view?usp=sharing

Model.h5 : https://drive.google.com/file/d/1xsO1iS3BiVvO1rICm56UEDAS3GNLSXgH/view?usp=sharing