

**Image classification using different methods of deep learning and computer vision methods of feature engineering to improve performances**

**Mathusan Kathirithamby1, Ali Akbar Sabzi Dizajyekan2, Majdi Saghrouni, Oussama Soukeur**

Department of Computer Engineering and Software Engineering, Polytechnique Montréal 2500 Chem. de Polytechnique, Montreal, Quebec, Canada

mathusan.kathirithamy@polymtl.ca1, ali-akbar.sabzi @polymtl.ca2

**Abstract**

*Convolutional Neural Network (CNN) is a type of deep learning artificial neural network that is extremely easy to set up yet remarkably efficient as an image classification algorithm. But the problem with deep CNN’s is that they are subject to the problem of the vanishing gradient. To solve this issue, CNNs skip-connection were invented with the purpose of solving the vanishing gradient problem. The two most popular ones are Residual Neural network (ResNet) and Densely Connected Convolutional Networks (DenseNet). Some of the problems with these types of networks is that we still apply convolutions across all channels which makes the models heavier [1].The answer to this problem is the Xception architecture which will be changing the regular architecture of CNNs by applying depth wise separable convolutions.*

**1 Introduction**

**1.1 Goals**

The main objective of this project is to classify the set of images contained in *CIFAR-10*. The dataset consists of 60,000 images grouped into 10 different classes with 6,000 images per class. Specifically, we chose 3 different architectures to conduct the classification. These architectures are convolutional networks that achieved good performance when classifying the *ImageNet* dataset, namely: *DenseNet, ResNet and Xception*.

Convolutional neural networks have a methodology similar to that of traditional supervised learning methods: they receive images as input, detect the features of each of them, then train a classifier on them.

By employing these architectures, it is essential to tweak the algorithms with different regularization and optimization techniques to get the most out of their performance. One of the techniques used is *SIFT*, which consists of detecting and effectively describing relevant features. Thus, we will have the opportunity to compare these architectures and confirm their performance obtained during the classification of the *ImageNet* dataset.

Our objective will therefore be, for a set of images, to obtain a maximum of exact estimates. However, we will not fall into the dilemma of executing everything in a reasonable time and with an acceptable consumption of resources since the data set is not large enough.

**1.2 Previous related work**

Several articles have focused on image classification using convolutional network architectures.

François Chollet [7] proposes a new deep convolutional neural network architecture that involves deep separable convolutions. This architecture is called *Xception* and it is based on another similar architecture but with a different operation: *Inception*. The architecture gives a top 1 accuracy of 0.79 thus exceeding the performance of other architectures mentioned in the research paper: *VGG-16*, *ResNet-152* and *Inception V3*.

In the paper, the author points out 2 minor differences between *Xception* and *Inception*. First, deep-separable convolutions apply spatial convolution per channel, then perform 1x1 convolution to compress the inputs, whereas in Inception the order is reversed. Second, the presence or absence of a nonlinearity after the first operation. While in Inception the 2 operations are followed by a ReLU activation function, the depth-separable convolutions in *Xception* are implemented without any nonlinearity.

He et al [2] introduce residual blocks that address the degradation of accuracy in the training set, in deep networks. Indeed, they demonstrated that with standard neural networks the error on the training set in a 20-layer network is lower than that of a 56-layer network. Thus, the network cannot generalize well for new data and becomes an inefficient model. This degradation indicates that continuing to stack layers does not necessarily improve model performance. Thus deep models will be difficult to optimize.

The main innovation for *ResNet* is the residual module (skip-connection). This module is specifically an identity residual module, which is a convolutional two-layer block with the same number of filters and a small size. The output of the second layer is added to the input of the first convolution layer.

The difference between *DenseNet* and *ResNet* is that the *ResNet* uses a summation of traits and skip-connections while DenseNet rather uses a concatenation of these aspects. Moreover, this makes *Densenet* have a higher capacity due to the connection of multi-layered functionalities superior to that of *ResNet*. However, this also requires more GPU capacity.

**1.3 Technique used and path followed**

In relation to the techniques covered in the course, we choose to exploit the idea of supervised learning. For this purpose, we have the CIFAR-10 dataset which contains 60,000 images grouped into 10 different classes.

Regarding learning, we used convolutional neural network architectures. These networks indeed represent a classic and effective choice for applications related to image recognition, due to their structure exploiting convolution operations well suited to this type of processing. In this context, we ran optimization techniques against these architectures and compared their performance.

The implementation of supervised learning is done in Python with the Pytorch library. Calculations are accelerated by GPU graphics processors.

**1.4 Computer vision methodology (SIFT)**

The main idea behind the SIFT algorithm is to detect interesting key-points (pixels), while making sure that the selected key-points are invariable to most of the known computer vision challenges. The list below provides such challenges and works that proves that SIFT algorithms can easily tackle such challenges:

Viewpoint: some objects can be detected wrongfully from different viewpoints Chen & Shang (2016).

Deformation: some objects may present deformations that make it harder for systems to detect them Zheng & Qian (2012).

Occlusion: represents the case where an object masks another one Zhou et al. (2009).

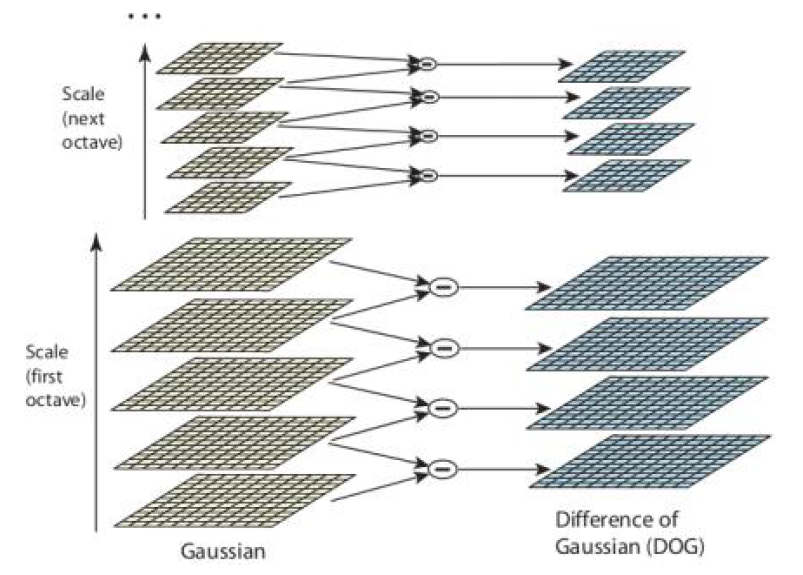
Illumination: represents the challenge induced by the change of illumination in the image Wang et al. (2015).

Texture: the textures are not easily detectable using SIFT and presents one of the challenges of such an algorithm.

Intraclass variation: also SIFT struggles in this particular challenge.

The SIFT algorithm is conceptualized in order to be scale invariant and immune to most of the computer vision challenges. This is achieved by four different steps.

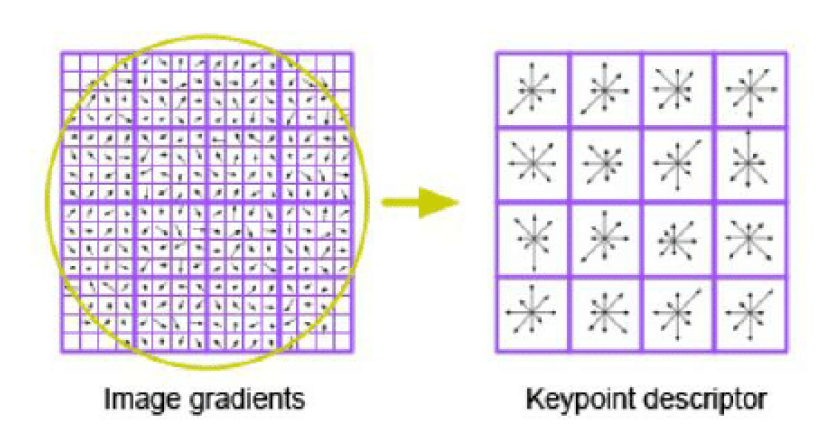
The first step is to provide a scale space (generate different octaves) combined with a Difference of Gaussian (DoG) and the calculation of the gradient (between the neighbors in the same octave and cross octaves) provides an elimination key-point detection mechanism. This achieves a feature extraction step with interest points that are scale invariant (different octaves) and resolution invariant (Gaussian blurring). This process is illustrated in figure 1.



**Figure 1**: Difference of Gaussians illustrated in LoweDavid (2004).

Second, refining the detected keypoints. At this step, three different types of keypoints are detected: edges, flat and accurate keypoints. The flat areas can be filtered by their intensity, for the edges, the combined use of Taylor expansion and the eigenvalues of the Hessian matrix with a threshold selection provides an accurate filter for the edges.

Third, the orientation and the scale of the gradient of detected keypoints is computed. The values as well as the bins of a created histogram (each bin 10 degrees) are taken if they surpass a certain threshold which thus creates a certain rotation invariance of the keypoint (Figure 2).



**Figure 2**: Key points extraction illustrated in Adel et al. (2014)

The last step is to calculate the keypoint descriptor. A neighborhood of (4x4) pixels is chosen and the keypoint descriptors represent the histogram (scale + orientation) of the keypoint with respect to the selected neighbor. This is typically calculated for an 8 bin histogram, so the resulting descriptor is of dimension 4x4x8 = 128. In our work, we only used the first three steps. In fact, we found more interest in the calculation of the SIFT to localize the keypoints. However, the values of the descriptors were discarded for more advanced features.

**4.Experiment:Resnet, Densenet and Xception**

**4.1 Data augmentation**

Three different techniques were introduced in our work to augment the data. First, we introduced resizing where we randomly changed the size of the images from (32,32) to the appropriate size based on the methods ((224,224) for ResNet and DensNet (299,299) for Xception). This helps augment the data by introducing more samples and creates a certain invariance by scale (same to the one introduced in SIFT). The second utilized technique is flipping. During this phase, the train images were randomly flipped (horizontally) in order to achieve a certain invariance by flips as well as augment the data size. Finally to push this even further, random rotations of 5 and 7 degrees with fixed degrees were introduced to achieve invariance by rotation.

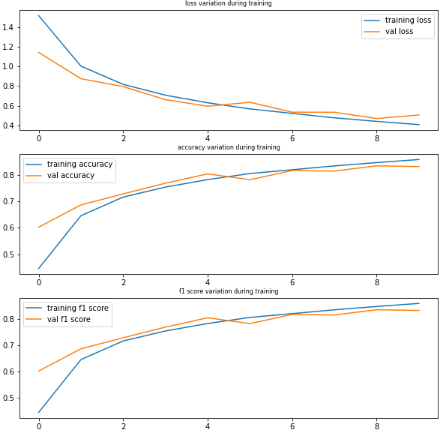
**4.2 Data preprocessing**

The normalization of the data was introduced by converting the pixels in the range [0,255] to [-1,1].

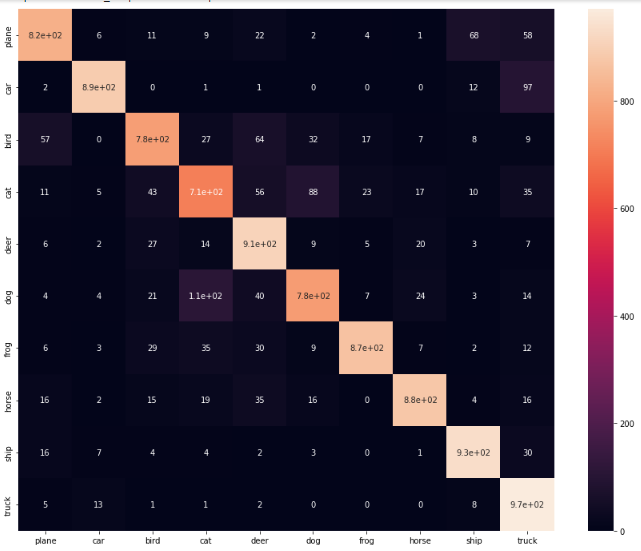
**4.3 Resnet**

Resnet-18 is a convolutional neural network with 18 layers, a kernel size of 7 for the first convolution and a kernel size 3 for the four sequential layers. We trained this model on our data set. After 10 epochs, the best validation accuracy was 85.26% with a f1-score equal to 85.26%. Figure 3 shows the evolution of the loss, accuracy and the f1 score for the two sets ‘ train & validation.the validation loss curve is decreasing then it starts to increase at the 8th epoch, same for validation accuracy and f1 score except that they increase then decrease.

As shown in the confusion matrix in Figure 4, the model correctly predicted the classes as the diagonale contains the greatest values other than the big number of cats that were misclassified as dogs. For this case, we can add more data for this class to our training data set to improve the performance of this class. As it is also shown in Figure 11, the model performed well with an average of 85% for f1 score and recall and 86% for precision.



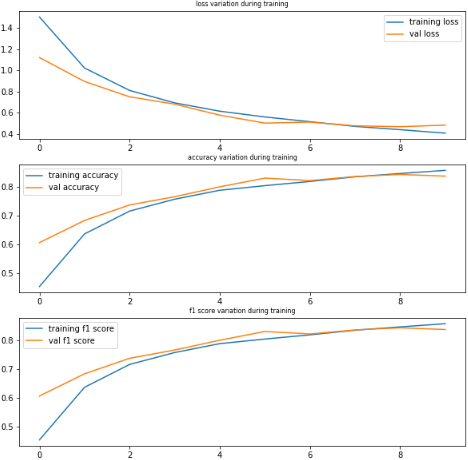
**Figure 3**: Resnet model plot on the validation and training dataset

****

**Figure 4**: Confusion matrix for the Resnet model

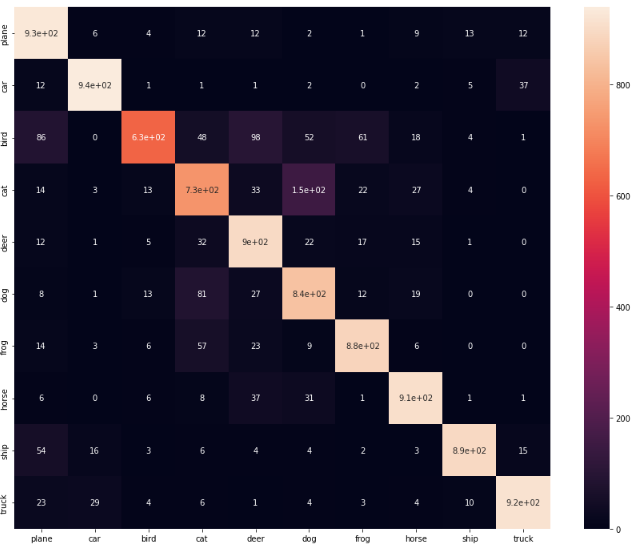
**4.54 Densenet**

Densenet is a convolutional neural network with 120 layers, a kernel size of 7 for the first convolution ,a kernel size 3 and 1 for the six dense layers and 3 transitional layers which contain 12 dense layers for the first transition, 24 dense layers for the second transition and 16 dense layers for the last one where each dense layer contain 2 convolution with a kernel 3 and 1. We trained this model on our data set. After 10 epochs, the best validation accuracy was 85.65% with a f1-score equal to 85.65%. For this model we notice a spike in the 5th epoch but the last result (after 10 epoch) is still the best as we can notice in Figure 5.



**Figure 5**: Densenet model plot on the validation and training dataset

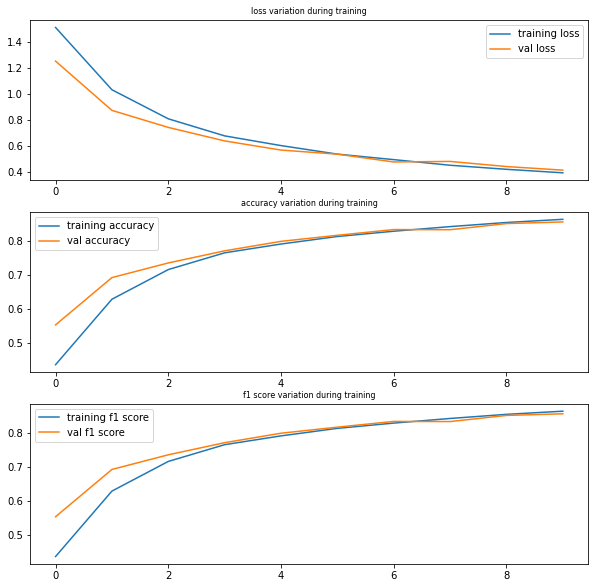
As shown in the confusion matrix in Figure 6, the model correctly predicted the classes where the diagonale contains the greatest values. But in this model, there are more dogs being classified as cats (inverse of Resnet results) as well as a high amount of cats still being classified as dogs. As it is shown in Figure 12, the model performed well with an average of 86% for f1 score, recall and precision. The class that was not well classified is still the class ‘CAT’ with a low recall value 73% and a low precision of 0.74 but also the class ‘DOG’ that has a precision of 0.75.



**Figure 6**: Confusion matrix for the Densenet model

**4.5 Xception**

Our Xception architecture has 3 different flows : Entry flow, Middle flow and Exit flow. The entry flow consists of 3 skip-connections, 6 depth wise separable convolutions, 3 max-pooling layers and 2 regular convolutions.The middle flow consists of a single skip-connection and 3 skip-connections, repeated 8 times. And the exit flow consists of a skip-connection, 4 depth wise separable convolutions, a max-pooling layer, a global average pooling layer and a fully-connected layer. As we can see this architecture is much more complexe compared to Resnet and Densenet. This is the model architecture that we built from scratch[6]. The validation f1 (accuracy) is 85.64% and the test accuracy/f1-score is 86.57% as we can see from the evolution of the accuracy and the loss in Figure 7. In the confusion matrix from Figure 8, we can clearly see that most of the classes were classified properly looking at the proportion of quantities in the diagonal of the matrix. But we still have the issue of the dogs being misclassified as cats from the 1.2e+02 amount visible in the confusion matrix which translates to a 75% precision for the class ‘DOG’ which is even lower than the 0.77 for the ‘CAT’ class as seen in Figure 13.



**Figure 7:**Xception model plot on the validation and training dataset

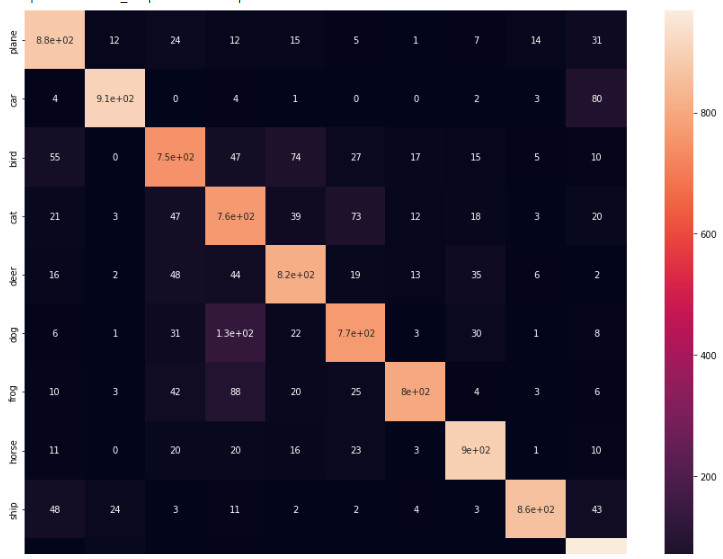


**Figure 8:**Confusion matrix for the Xception model

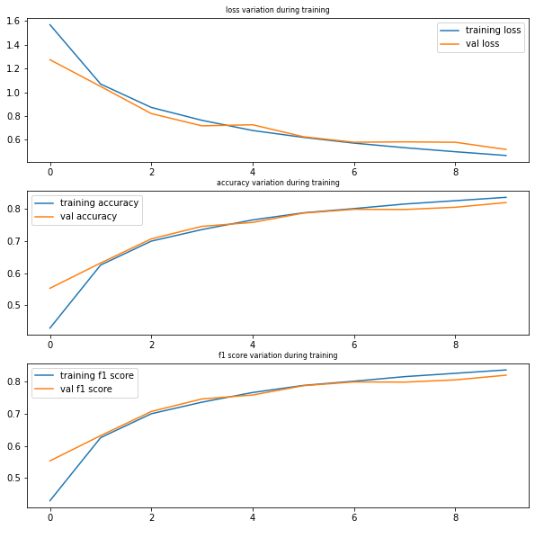
**4.6 SIFT+Xception**

The hyperparameters of the SIFT algorithm were fine tuned to provide the best mask and cropped images. We mention particularly, “nfeatures” which is the number of features extracted from the image, “nOctaveLayers” which represents the number of octave layers (how many times we are scaling the images to obtain scale invariance), “sigma” which is the standard deviation of the used Gaussian which rules over the blurring factor of the algorithm and finally contrastThreshold which is the selectivity of the SIFT model. Concerning the mathematical morphology operator, the only hyperparameter is the number of times the closing is going to be applied. This was also manually selected to provide close enough geometries; the selected value is 20.

Adding SIFT to Xception did not improve the performance of the model. The model reached an f1-score of 0.82 in the validation set and 0.84 in the test dataset. The accuracy of the model for “Bird”, “Cat” and “Dog” class is still low.

****

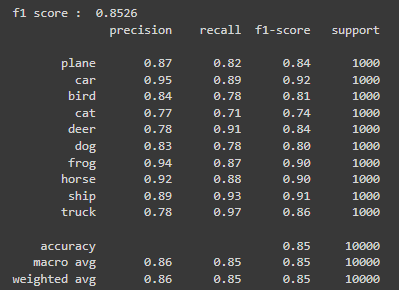
**Figure 9:**Confusion matrix for the SIFT+Xception model

****

**Figure 10:** SIFT+Xception model plot on the validation and training dataset

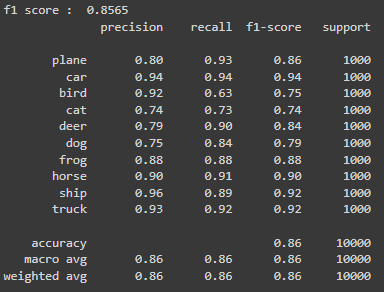
**5. Critical Analysis**

From the Figure 3,5 and 7 we can clearly see that the training and validation loss, accuracy and f1 score are neck and neck in the Xception model compared to the other two, which can indicate that with more epochs, we can definitely get an improvement in the Xception architecture without facing overfitting issues.

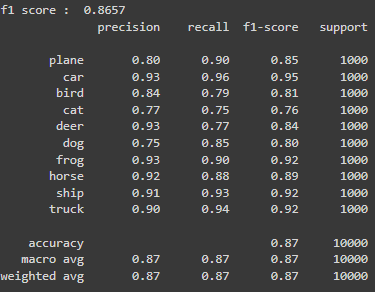


**Figure 11:**Precision, Recall, F1-score and support for the Resnet model

As mentioned in the Experiment part, all models have issues of misclassification for the classes of ‘CAT’ and ‘DOG’ as we can see in Figure 11,12 and 13. Resnet has the lowest accuracy of model with an f1-score of 85.26% on the test data but it has a precision of 0.77 and 0.71 for the recall of the ‘CAT’ class. For the Densenet model , we get a reasonable 85.65% f1-score for the test dataset, but we get a 0.74 as precision and 0.73 for the recall for the ‘CAT’ class. Similarly, for the Xception model we see that the ‘CAT’ class has a precision of 0.77 and a recall of 0.75. We can see that as the model complexity is increased (resnet being the least and Xception being the more complex one), we can see that the f1-score for the ‘CAT’ class is what is improved the most with the following f1-scores: 0.74, 0.74 and 0.76 for Resnet, Densenet and Xception respectfully.



**Figure 12:**Precision, Recall, F1-score and support for the Densenet model



**Figure 13:**Precision, Recall, F1-score and support for the Xception model

The primary objective of running several models was to do a majority voting to determine the image label, however since all models have difficulty classifying special classes, majority voting does not seem to improve overall performance.

As students, we only used the given GPU’s assigned to us with our Google Colab Pro membership (K80, T4 or P100) with 32GB of RAM. Therefore, we can assume all our models were trained with very similar setups. A key difference between our models was the time it took us to train it. As the model and architecture complexity became higher, we also got higher runtimes. For 10 epochs, Resnet took 1 270 seconds (approx. 21 min), Densenet took 3 482 seconds (approx. 58 mins) and Xception took 12 952 seconds (approx. 4 hours). So even if we do get a 1% increase in the test data for the Xception model compared to Densenet, it comes at a cost of almost 3 more hours (3.7 times the runtime). The densenet-121 model had much more layers compared to the resnet-18, which also explains the difference in runtime between the two models.

**6.Conclusion**

This also goes on to explain why Resnet and Densenet models are highly popular models for many computer vision tasks as it would also be our choice given the amount of runtime Xception takes. They have very similar results to the Xception model but with much less complexity and much less runtime needed. But for purely better accuracy, the Xception model seems to have a slight improvement over the other two just like Francois Chollet [7] mentioned in his article on his own dataset.

**References**

[1] Yong Chen and Lei Shang. *Improved sift image registration algorithm on characteristic statistical*

*distributions and consistency constraints.* Optik, 127(2):900–911, 2016.

[2] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift.

*Computer vision and image understanding*, 113(3):345–352, 2009.

[3] Yu Wang, Xiaojuan Ban, Jie Chen, Bo Hu, and Xing Yang. *License plate recognition based on sift feature*. Optik, 126(21):2895–2901, 2015.

[4] G LoweDavid. *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision, 2004.

[5] Ebtsam Adel, Mohammed Elmogy, and Hazem El-Bakry. Image stitching based on feature extraction techniques: *A survey. International Journal of Computer Applications,* 99:1–8, 08 2014. Doi:10.5120/17374-7818.

[6] Sik-Ho Tsang (reference for our code), *Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)*, 2018.

[7] Francois Chollet. CholletXception: Deep Learning with Depthwise Separable Convolutions, 2016