Image Classification: CIFAR-10, Convolutional Neural Networks

Candidate: 020845

1. INTRODUCTION

Convolutional Neural Networks (CNNs) have dominated computer vision classification problems since 2012 when AlexNet was submitted, and was the winning entry, to the annual ImageNet competition [1]. CNNs are a type of artificial neural network which make use of convolutional layers allowing them the capacity to learn the large amount of information required for hard image recognition tasks. This report describes experimentation in the design and implementation of CNNs for classification. For this task the CIFAR-10 data set was used, which consists of 60,000, 32×32-pixel colour images. The image set is equally balanced between the 10 classes meaning each class has, in total, 6,000 images. The classes are: airplane, car, bird, cat, deer, dog, frog, horse, ship, truck. An example of the images can be seen in figure 1. A baseline CNN (BASE) was provided and its performance would act as a baseline for comparison to models which would be built during the experiments. The challenge is to construct *improved* models (IMP) and gain a knowledge and understanding of CNNs in the process.

Figure 1. Examples of images in CIFAR-10

2. EXPERIMENTAL

The CIFAR data set comes pre-split into 50,000 images for training and 10,000 for testing. However, for our purposes the test set was used more of as a validation set and a custom test set of 200 images was constructed. Because of this we will refer to the CIFAR-10 test set as *validation* set and the custom test set as *test* set from now on. Images for the test set were taken from various websites on the internet (figure 2). This consisted of square images of various sizes which were then resized to 32×32 pixels. Each class was represented by 20 images.

Two improved models, IMP1 and IMP2, were constructed using the same architecture (Table 1), however, IMP2 was trained using image augmentation and IMP1 was not. The image augmentation was implemented using Keras' *ImageDataGenerator* object with the settings enabling a 20 degree rotation, positional shift of 0.1, a zoom increase of 0.2 and decrease of 0.3, channel shift range of 0.2 and horizontal flip. The optimization algorithm used was Adam and although this is an adaptive gradient method, we did experiment with decreasing the learning rate manually. Each convolutional layer used L2 regularization with a weight decay of 1×10^{-4}.

Two instances of MobileNet were trained, *MNET1* and *MNET2*. For this, the pretrained ImageNet weights were loaded and the input layer was replaced with a tensor of dimensions 128×128. Image augmentation was used to train both networks using the same parameters discussed earlier. Every layer in MNET1 was locked as to prevent further weight adjustment. However, an additional dense layer with softmax activation function was added to serve as the output which was a vector of size 10. MNET2 was trained in the same way except that only the layers which come before the last convolutional layer were locked to allow the last convolutional layer to be trained. Furthermore, a dropout layer of 0.4 was added before the output layer.

Table 1. Architecture of implemented CNNs

BASE		IMP	
Layer type	Output shape	Layer type	Output shape
Conv. 2D	32, 32, 32	Conv. 2d	32, 32, 32
Batch norm.	32, 32, 32	ELU	32, 32, 32
RELU	32, 32, 32	Conv. 2d	32, 32, 32
Flatten()	32768	ELU	32, 32, 32
Dense	512	Max pooling 2D	16, 16, 32
RELU	512	Dropout (0.1)	16, 16, 32
Dropout (0.5)	512	Conv. 2d	16, 16, 64
Dense	10	ELU	16, 16, 64
Softmax	10	Max pooling	8, 8, 6
		Dropout (0.2)	8, 8, 64
		Conv. 2D	8, 8, 128
		ELU	8, 8, 128
		Conv. 2D	8, 8, 256
		ELU	8, 8, 256
		Max pooling	4, 4, 256
		Dropout (0.3)	4, 4, 256
		Flatten()	40970
		Dense	10
		Softmax	10

Figure 2. Random sample of unfiltered custom test set images

Some visualizations were implemented using two different methods to try and gain some insight into how the CNNs work. The first method uses an image as input and sets a chosen layer as output. For this we selected a small sample of activation layers.

The CNN's predict method is used to collect output from one of the activation layers, which contains many 2D filter arrays, each of which is turned into an RGB image by using a colormap. Because some activation layers can contain hundreds of filters, we took a sample of 16 and tiled then into a 2×8 grid. This is repeated for each selected activation layer in the network and each grid is stacked so that tiles at the top are nearest the input and tiles at the bottom are nearest the output (figures 1B and 2B). The second visualization uses a method similar to Google Deep Dream which can be found here [2]. This process does not use an input image per se, but rather feeds the input layer into the model by gradient ascent and uses loss to model the gradient in a chosen convolutional layer. The filters with the highest loss are selected, as these are likely to have more interesting topologies. These are tiled together to make an RGB image (figures 1C and 2C).

3. RESULTS AND DISCUSSION

It was noticed through trial and error that learning rate increased and classification performance improved with the use of the Exponential Linear Unit (ELU) activation function rather than RELU. In contrast to ReLUs, ELUs have negative values which allows them to push mean unit activations closer to zero which is similar to batch normalization, however, the computational cost is less. Mean shifts toward zero speed up learning by bringing the normal gradient closer to the natural gradient because of a reduced bias shift effect [3]. Furthermore, experimentation with learning rate showed that decreasing the learning rate at certain points sped up convergence somewhat. This can be seen in figure 3 plots for IMP 1 and 2 where the there are sudden rapid increases in accuracy which look like humps in the

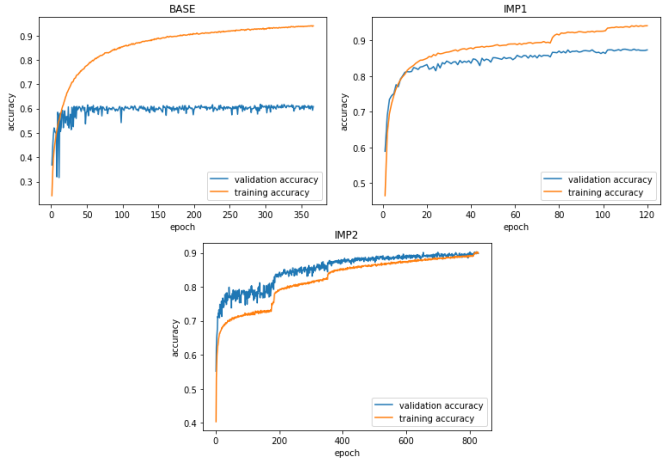

Figure 3. Accuracy during training of BASE and IMP models

The IMP models both outperformed the BASE model by a considerable amount (above figure). The BASE model yielded a comparatively low accuracy score, which can be attributed to two reasons. Firstly, the network did not have enough parameters to learn the training data effectively. This is evident as adding more layers as with IMP1 has increased the network's learning capacity, which has increased the validation accuracy. Secondly, the training set was too small, which was remedied by using image augmentation. This has allowed the validation accuracy to improve even further albeit a small amount (table 2). It was noticed that, although image augmentation can offer improvement, it can give worse results if the augmented images not are representable of the population that is being modelled. For example, shifting the object in the image so that it is no longer in the image will force the model to learn features that are not necessarily attributed to the object. i.e. background textures and objects. Furthermore, convergence takes longer. In this experiment we only implemented relatively minor augmentations on IMP and the convergence took roughly 8 times more epochs. This may indicate that the IMP network required more parameters in order to learn the extra information that image augmentation provided.

MNET1 managed to obtain 73.1 and 61.5 % accuracy on training and validation sets respectively which is impressive considering that it required very little work. It can be seen in the figure below that the model converges rapidly at 4 epochs but badly overfits. This highlights partly how CNNs operate, in that MNET has already learned very well the lower level features of classes such as these. These features are transferable to other applications. A lot of what is required after that, is mapping those learned features to the labels we are classifying. Allowing the last convolutional layer to be trainable as with MNET2 saw a large improvement in performance (table 2). MNET2 took longer to converge in comparison (30 epochs), nevertheless, this is still much quicker than with the IMP models. Looking at the figures below it can be seen that MNET2 was still over fitting, however, it is quite possible better performance is obtainable with some modification such as dropout, more image augmentation or allowing more layers to be trained.

Table 2. Performance comparison of implemented CNNs

	Validation	Test	Test (filtered)
BASE	60.7	30.5	47.0
IMP1	87.3	54.5	76.0
IMP2	88.4	58.5	79.0
MNET1	73.4	61.5	69.0
MNET2	81.2	75.0	75.5

Accuracy (%)

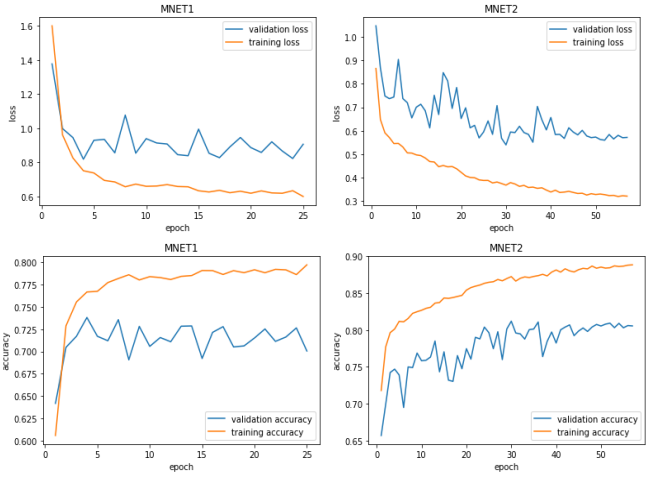

Figure 4. Training performance of MNET1 and MNET2

Square images were purposefully selected for the test set, as the input sizes of the CNNs were square, and this avoided complicated resizing methods. Once a test set had been constructed and appropriately resized, it was aparent that there appeared to be a difference in the sharpness of the images, in comparison to those in the CIFAR-10 set. The test set images were sharper furthermore,

the sharpness of the CIFAR-10 images appeared to be uniform throughout the set, suggesting that either filtering had been applied using a kernel size dependent on image size, the original images were 32×32 or the original images were at least the same original size. suggesting that if a kernel filter was used the kernel size of the filter, that was used, was dependent on the original image size. The accuracies obtained on the test set images were much lower than those obtained on CIFAR-10 (Table 2) and this difference in sharpness could be a contributing factor. To investigate this, we applied a gaussian blur with a kernel size dependent on image size and attempted to qualitatively match that of the CIFAR-10 set. Where w is the original image width, the kernel size was $s \times s$, where s is the nearest integer obtained by $s = w / 75$. What is interesting is that, with the unfiltered test set, the MNET models performed slightly better, despite having much lower accuracies on CIFAR-10. This shows that, although the MNET may not have fit the training data as well as those that were trained from scratch, the MNET models were more robust. Those trained from scratch on CIFAR-10 show more dependency on the sharpness of the image which can be seen in Table 2 where their performance greatly improved when we applied the filter.

The classification results for IMP2 and MNET2, with images of the misclassified samples can be seen in figures 1C and 2C. One similarity between the two CNN models is that they both have trouble classifying dogs. This is understandable as there is much variation in dogs [4], so there are a lot of features to learn and these features are not necessarily always present for different breeds of dog. Furthermore, there is likely to be much overlap in features with the other similar mammals in this data set. Moreover, there are relatively fewer misclassifications of other classes being classed as dogs. Another common misclassification is that of cars as trucks. Additionally, it is apparent that the misclassifications do not occur the other way suggesting that both networks have a bias towards trucks. It could be argued that there is more variation in dogs and cars due to there being many very different breeds and models, than there is for the other classes. The results show that in this case, these classes get classified as other classes rather than the converse.

Comparing the visualizations of activations in IMP2 and MNET2 in appendix A, it can be seen there are some noticeable differences. Firstly, there are some filters which are completely black. This is attributed to those filters not being activated. This means that they respond to features which are not present in the image. There are more black filters in MNET2 than IMP2 and it can be seen that some of these filters are black for each input image. They may respond to features which are not even in the images in this experiment. IMNET2 is a much larger network than IMP2, with some 14 convolutional layers in comparison to 6 in IMP2, and it is likely that there is much redundancy. Another notable difference is that the activations in the early layers in MNET2 look much higher in resolution. This means they are capturing more information, however, the input sizes and dimensions of these layers are sometimes the same. The difference in resolution must be attributed to how MNET was originally trained, which used higher resolution images. The results in this experiment so far suggest that it is better to train on higher resolutions rather than lower resolutions if a choice must be made.

The visualizations in appendix A shows how information is propagated through the network. The early layers seem to be zooming in on certain regions of the image. Deeper into the network the regions are broken down into features such as edges and corners. As we approach the output, the filters encode more abstract/ higher-level information which is more associated with the class of the image and less relatable to the image. The visualizations in figure B support this observation. The earlier layers show much finer and simpler textures. Often fine matrices of squares and fine lines. Whereas the deeper layers encode more complex textures. The MNET2 filters seem to encode much more complex information than that of IMP2 especially in the deeper layers. This can be seen by the higher complexity in the topology of those images. In addition, the filter for MNET2 change more as we go deeper through the network, in comparison to IMP2 where they do not change much. There is greater change as information travels through the network. This evidences that MNET2 has learned higher level information. Furthermore, the IMP2 filters are more homogeneous which suggest that it is less able to turn the simpler concepts into more complex concepts.

4. CONCLUSION AND FURTHER WORK

The experiments have highlighted the usefulness of pretrained networks that have made use of hard to obtain computational and time resources. These networks allow powerful and robust performance.

Looking at figure 4, it is likely that further image augmentation could increase the validation accuracy further. More experimentation with image augmentation could yield better performance with MNET2. It is unlikely that it needs more layers as it is already learning the training data well.

Another visualization method that would be interesting to use would be occlusion mapping or class activation mapping [5]. The objective here is to determine which parts of the image the CNN is focusing on. This can be done by randomly setting parts of the image to zero and applying it as input to the CNN. It is possible to make a 2D visualization which is a function of the change in probability of the correct class as different regions are occluded. This can identify whether the CNN is learning background features such as trees or water instead of the object of interest.

5. REFERENCES

[1] Krizhevsky, A., Sutskever, I., and Hinton, G., 2012. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*.

[2] Chollet, F. 2016. https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html.

[3] Clevert, D.U., T. Hochreiter, S., 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *CoRR*.

[4] Schaffer, A.L., Irion, D.N., Eggleston, M.L., Pedersen, N.C., Hughes, S.S., and Famula, T.R., 2003. Analysis of Genetic Variation in 28 Dog Breed Populations With 100 Microsatellite Markers. *Journal of Heredity 94*, 1, 81-87. DOI= http://dx.doi.org/10.1093/jhered/esg004 %J Journal of Heredity.

[5] Sáez Trigueros, D., Meng, L., and Hartnett, M., 2018. Enhancing convolutional neural networks for face recognition with occlusion maps and batch triplet loss. *Image and Vision Computing 79*(2018/11/01/), 99-108. DOI= http://dx.doi.org/https://doi.org/10.1016/j.imavis.2018.09.011 .

APPENDIX A

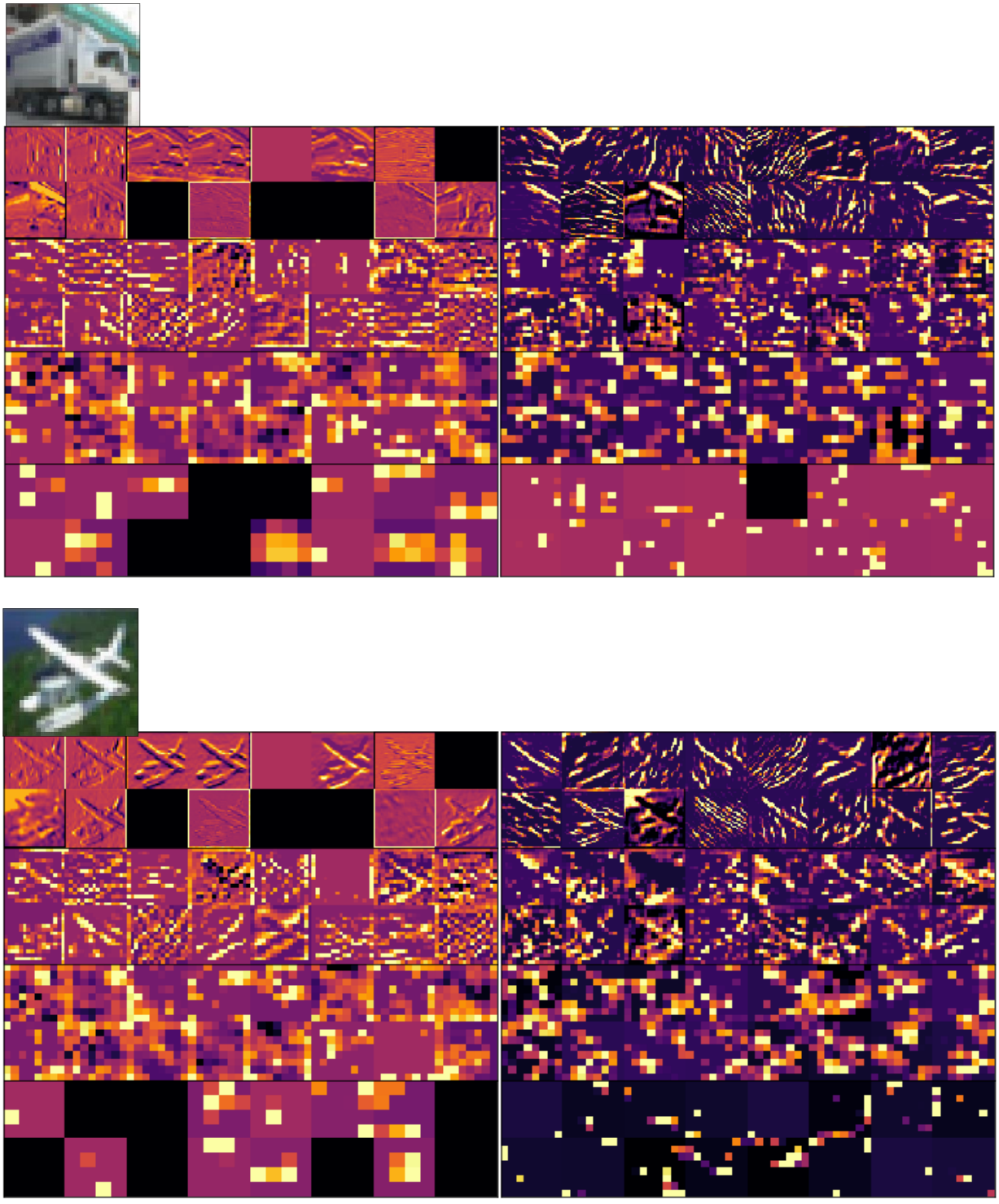

Figure 1B. Comparison of activations in MNET2 and IMP2 from truck and airplane images

Left: MNET2, **Right:** IMP2. Each quadrant contains 4 layers moving from input direction to output direction. Each layer is a sample of 16 filters from an activation layer. A sample was taken as the activation layers towards the output end of the network can have too many slices to fit on a page. The activation layers taken from roughly evenly distributed locations through the network.

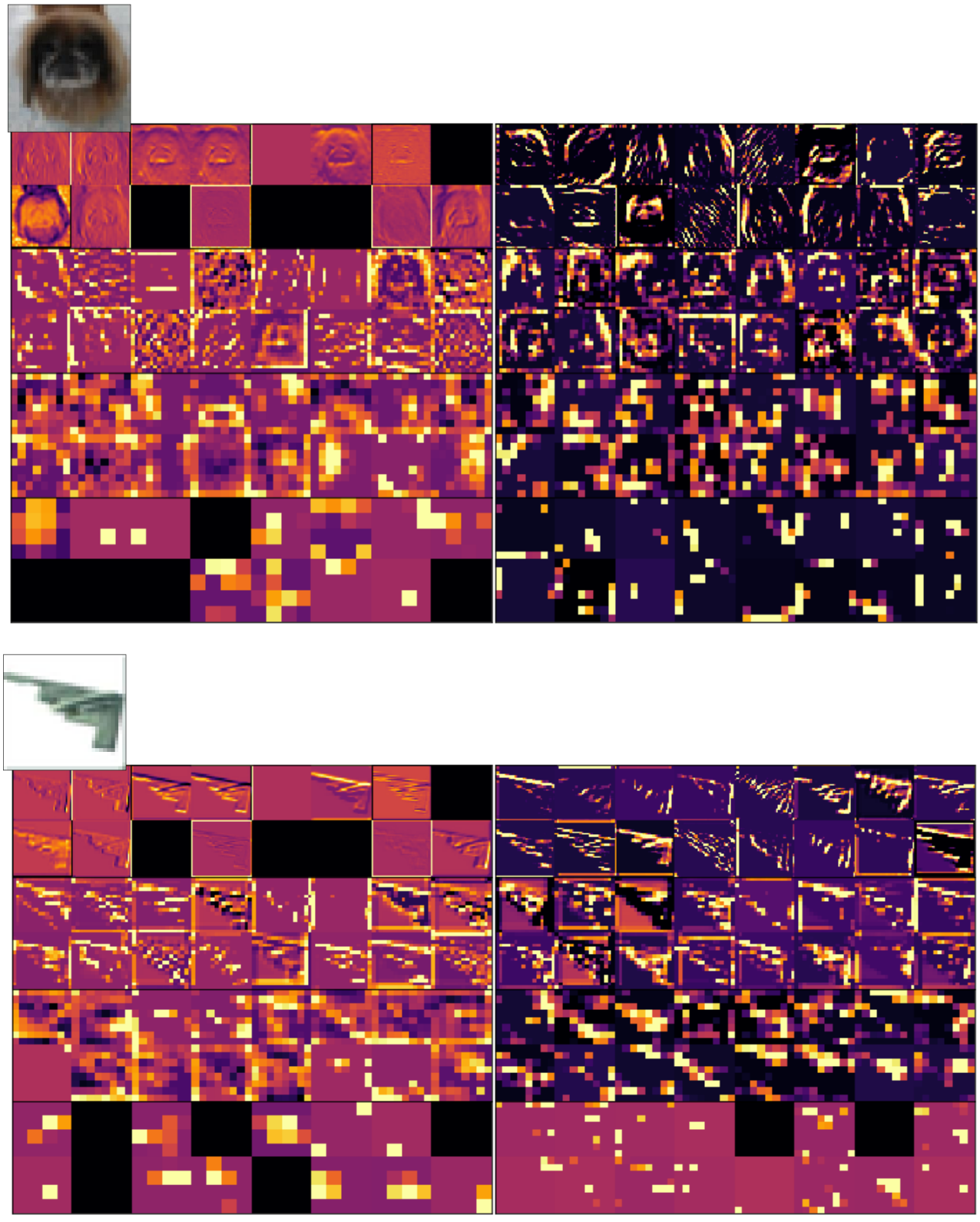

Figure 2B. Comparison of activations in MNET2 and IMP2 from dog and airplane images

Left: MNET2, **Right:** IMP2. Each quadrant contains 4 layers moving from input direction to output direction. Each layer is a sample of 16 filters from an activation layer. A sample was taken as the activation layers towards the output end of the network can have too many slices to fit on a page. The activation layers taken from roughly evenly distributed locations through the network.

APPENDIX B

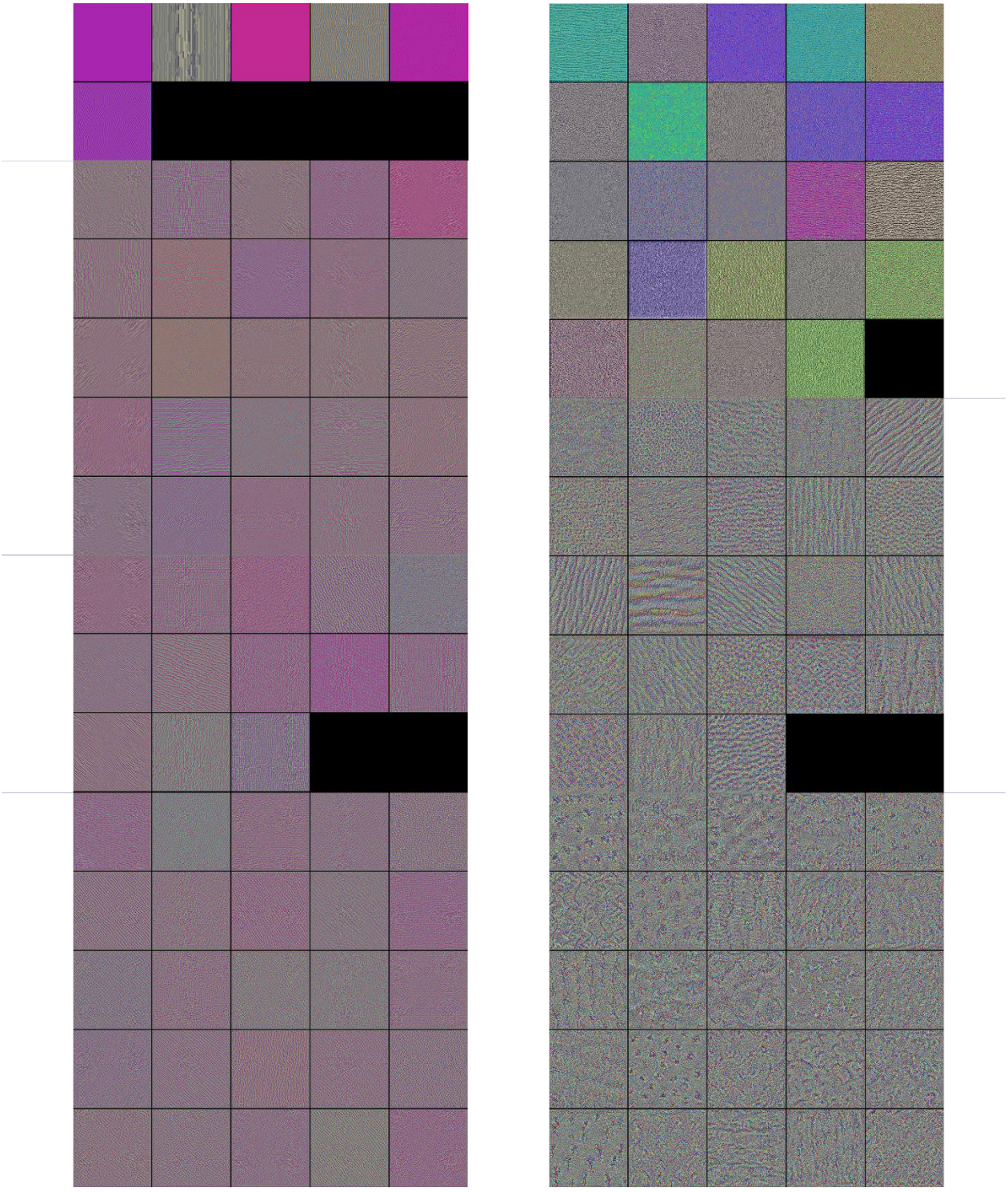

Figure 1B. Google Deep Dream style visualization of IMP2 and MNET2

Left: composed of filters from 4 Elu layers of IMP2, **Right:** composed of filters from 3 ReLU layers of MNET2. Thin blue lines have been used to distinguish divisions between different layers. Layers nearest the top are nearest the input and layers nearest the bottom are nearest the ouput.

APPENDIX C

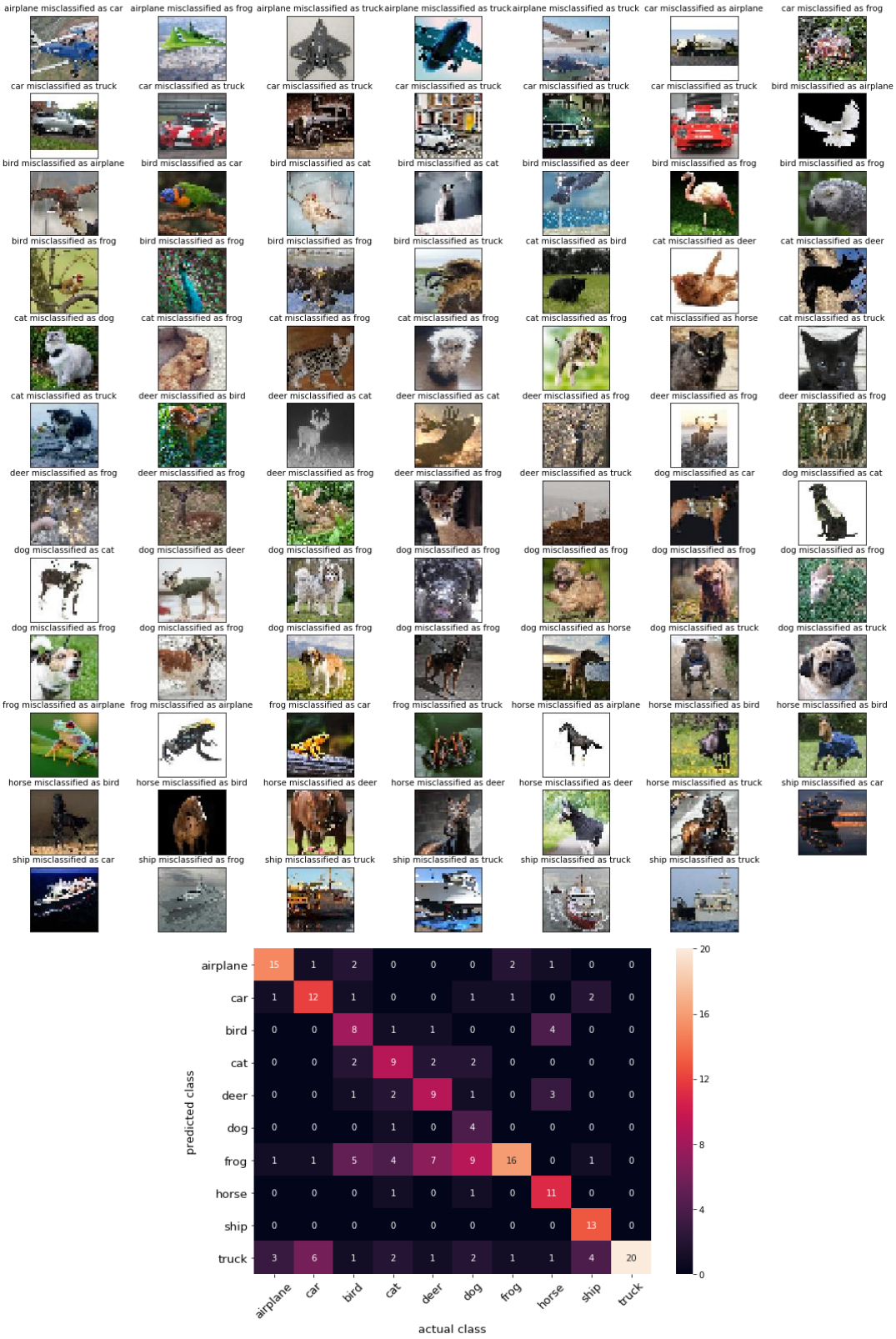

Figure 1C. Classification results for IMP2 on custom test set

Top: misclassified images, **Bottom:** confusion matrix of all classifications.
These results were obtained working with unfiltered images.

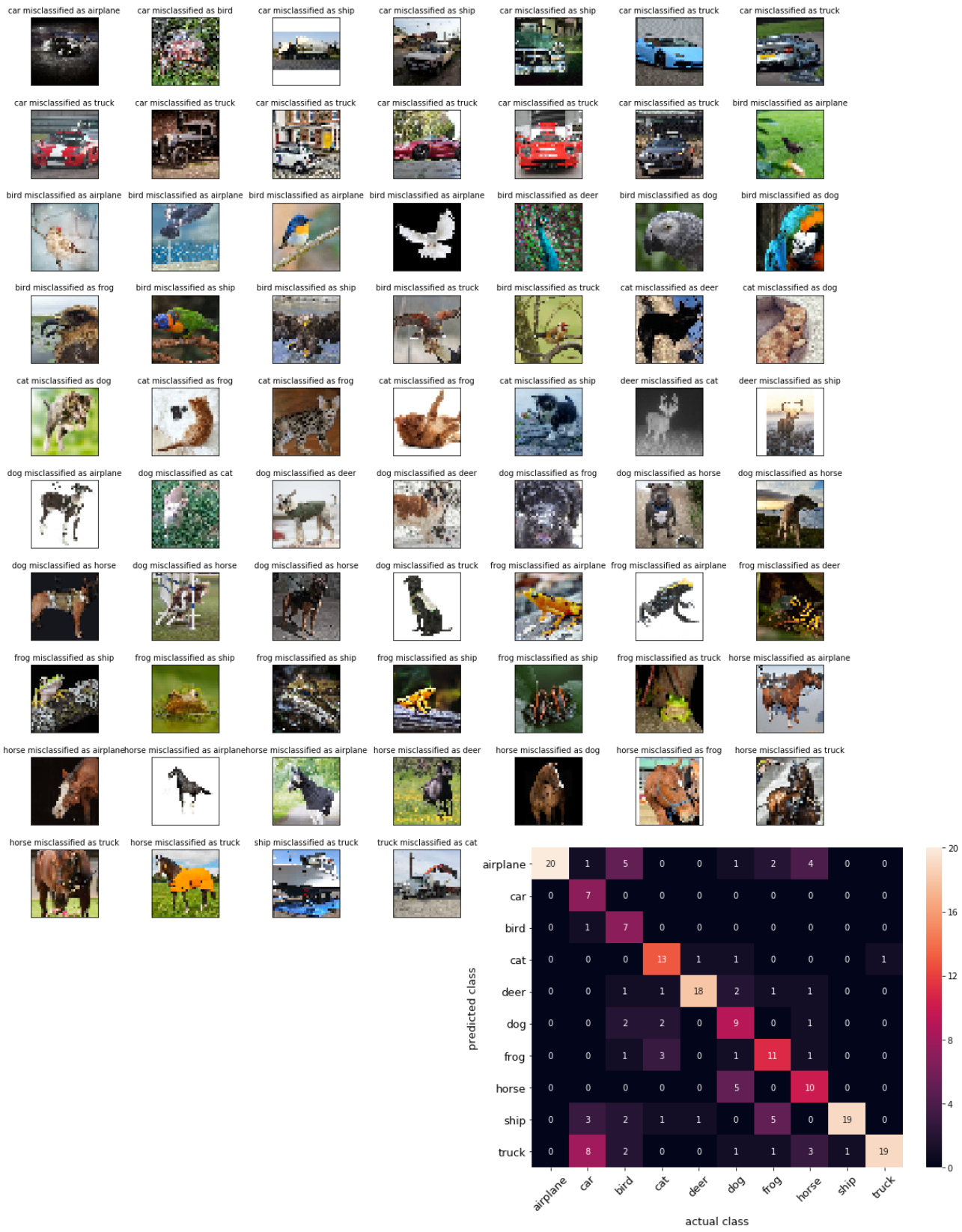

Figure 2C. Classification results for MNET2 on custom test set

Top: misclassified images, **Bottom:** confusion matrix of all classifications.
These results were obtained working with unfiltered images.