Fine-grained vehicle classification on real traffic surveillance

Martin O'Donnell

Abstract—**Fine-grained vehicle classification is the process of identifying a vehicle by its' make, model and year. This type of classification can be beneficial in an intelligent transportation system. There has been substantial research on classifiers that can detect fine-grained details on high-resolution vehicle images, but this does not represent the real-world data. Most surveillance and traffic cameras create lower resolution images, resulting in less detailed features. This problem, along with pose and viewpoints issues, leads to a difficult recognition challenge. This paper proposed several deep learning approaches to boost the performance on the low-resolution boxcar's dataset. The first approach transfers knowledge from one dataset to another, obtaining an accuracy of 84.9% when using datasets with the same resolution. The second approach makes use of multitask and auxiliary learning to improve the learning of the multiple labels involved in fine-grain identification. Multitask learning models obtained an accuracy of 83.8% when the tasks are learnt individually and reconstructed into the fine-grain label. Auxiliary learning boosts the performance to 85.3%. Combining knowledge transfer and auxiliary learning approaches increases the accuracy further to 86.3%. Channel max pooling is also validated, boosting the performing by 0.4% on the low-resolution dataset**

Index Terms—**Fine-grained vehicle classification, low-resolution, knowledge transfer, multitask learning, auxiliary learning**

I. INTRODUCTION

Fine-grained vehicle classification is the task of identifying a specific vehicle using its make, model and year. This classification problem can be quite a challenging task because the visual differences between vehicles can be small. A small change in viewpoint, pose, or location of the object in the image can overwhelm a model, causing an incorrect prediction. Compared to a classification problem such as differentiating between car types, a fine-grained vehicle classifier needs to take in a wider range of unique properties in an image to try to identify a single-vehicle. This task is still challenging for humans to do themselves and generally requires extensive domain knowledge.

The biggest challenge for fine-grained classification is the ability to cope with large inter-class and intra-class variations. There are instances where vehicles with the same colour, viewpoint and appearance appear with different makes, models and years. This brings about low inter-class variation. High intra-class variation can also be seen with vehicles that have the same make, model, and year but different colours or viewpoints. This challenge is heightened when lower resolution images are classified. The majority of applications such as surveillance or traffic cameras currently in use, output low-resolution images. These lower resolution images are representative of data from a real-world application and pose an extra challenge to the fine-grained classification. The challenge of distinguishing between the inter-class and intra-class variation increases with lower-resolution images. Features that would be easily identifiable in higher-resolution images are not in lower-resolution images.

This paper will investigate the current state-of-the-art fine-grained vehicle classification for low-resolution images. The paper proposes a deep learning approach to transfer knowledge from one dataset to another. Multitask and auxiliary learning models will also be proposed to improve fine-grained classification further. Experiments are conducted to investigate if combining these approaches results in a better overall performance.

II. RELATED WORK

Before deep learning approaches, to accomplish fine-grained classification part detection was used. Part detection approaches primarily focused on the front and rear of vehicles because most distinguishing features are found there. Authors of [1] and [2] look at detecting the license plate of vehicles. They use the license plate as an anchor and find regions of interest around it. These regions are input into a classifier to recognise the make and model. Authors of [3] and [4] look at extracting standard features such as headlights from the front and rear. These techniques reduce intra-class variation. Visually similar cars will have slightly different parts, and this can be picked up with these approaches. These approaches do not deal with pose variation. The front or rear view must be in focus and in-frame for these methods to work.

To address pose variation, 3D-model based approaches were proposed**.** Krause et al. [5] proposed to use 3D CAD models to train geometry and viewpoint classifiers. This brought about improvements of 3D versions of Spatial Pyramid and BubbleBank [6] by 3D patch sampling and rectification. The use of CAD models improves the detection of vehicles with multiple viewpoints and poses but involves

[M]artin O'Donnell is with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK (e-mail: modonnell24@qub.ac.uk).

knowing the 3D CAD models before training. If this technique were transferred to another dataset, additional CAD models would need to be generated to ensure there was a match for each vehicle in the dataset. The paper uses 41 models for 196 different vehicles so some vehicles can use the same model.

The success in convolutional networks on large-scale classification demonstrates that powerful features can be extracted from each pixel in an image. Papers such as [7],[8] and [9] have taken the deep learning models that have achieved high accuracies on large scale image classification tasks and tested them on fine-grained vehicle datasets. Models such as VggNet, DensetNet and GoogleNet have been tested and shown excellent performance when trained from scratch or fine-tuned. The GoogleNet model, that was pre-initialised from the ImageNet trained weights, outperforms some traditional approaches for fine-grained recognition such as part-based approaches.

The advantage of using a deep learning approach is that the model can learn distinguishing features across the vehicle to identify it. The downside is that significant data is required to train the models. Since deep learning approaches have proven to work on higher resolution images, more datasets that represent real-world data have been obtained. Liao et al. [3] created a dataset of 1,482 vehicles with eight classes, but this is too small to train a deep learning model. The CompCars dataset [9] contains both high and low-resolution images. It contains 136,726 web-based images and 50,000 surveillance-based images. These surveillance-based images are captured from the front view. This dataset has been used by many papers to propose deep learning approaches, but they can only identify vehicles from the front. [10]–[12] proposed new datasets from real-world scenarios but they are used for vehicle tracking. They do not contain annotations for fine-grained classification, and it would need considerable manual work and expert knowledge to label them in fine-grained detail. The BoxCars dataset [13] contains surveillance camera images and could be used to train a deep learning model. This dataset includes 116,286 images of vehicles with fine-grained labels from multiple fixed surveillance cameras. The cameras take multiple photos of the same vehicle from multiple viewpoints and poses. This is one of the first papers to propose a novel low-resolution dataset capable of fine-grain classification. Before these datasets, the norm was to downsample high-resolution datasets to train neural networks for low-resolution classification. [14]

Sochor et al. [13] use the BoxCars dataset to propose a deep learning approach which decreases the classification error by 26% and boosts verification average precision by 208%. This paper's proposed solution starts by finding the 3D boundary box of each vehicle before training. [15] This boundary box is used to unpack the vehicle image, create a rasterised low-resolution shape and information about the 3D vehicle orientation. All this information is input into a convolutional network for each vehicle at training and testing. Using the 3D boundary box, this approach can deal with pose and viewpoint variance. It can also adapt to new models of cars on its own. It not only recognises the pre-trained set of models but can also verify whether two given vehicle samples are of the same make and model or not, without previously seeing these particular vehicle types.

Sochor et al. [16] extended the research from their last paper [13] by proposing a way to automatically estimate the 3D boundary box at training and test time. This method doesn't perform as well as their last method but allows a boundary box to be generated when a precise construction is not possible. The paper also proposes several novel data augmentation techniques. They alter the colour of the image and block out random parts of the vehicle using noise. These additions are randomly added, during training, to help improve the diversity of the training samples, giving a more robust model. Using the new features proposed, the paper goes on to test how each feature affects the performance on several state-of-the-art networks. It was found that VGG-16 model performed best when implemented with all the improvements.

Several papers have proposed approaches to fine-grained recognition using higher-resolution images that will be discussed below. These may be transferable to help solve fine-grained recognition on lower-resolution images. Ma et al. [17] propose adding a channel max-pooling scheme between the fully connected and convolutional layer. This leads to extracting more discriminate features due to smaller feature maps. It also has the advantage of decreasing the number of parameters, in turn, reducing the model complexity. Hu et al. [18] proposed a spatially weighted pooling which improved the robustness and effectiveness of feature representation of most dominant CNNs. This approach was based on the CompCars dataset and showed promising results against the hybrid dataset. Li et al. [19] focus on the loss function of the neural network, proposing a new function called Dual Cross-Entropy Loss. This new loss function places constraints on the probability that a data point is assigned to a class other than it's ground-truth class. This can alleviate the vanishing of the gradient when the value of the cross-entropy loss is close to zero. The authors of [20] and [21] propose multitask convolutional networks when working with vehicle datasets. [20] research is evaluated on a model's performance to retrieve images of the same type. They found that when coarse and fine-grained labels were used with appropriate loss function, the model retrieval accuracy would improve from other state-of-the-art results. [21]'s goal is to classify a vehicle type and colour. They again found that combining both these tasks into one model outperformed single-task classifiers and sped up execution time.

III. Technical Section

A deep learning approach is proposed in this paper to improve the classification of low-resolution images in fine-grained detail. The first approach used, builds on the research from [22] to experiment how knowledge can be transferred from high-resolution data to low-resolution data. This paper builds on their research to investigate if knowledge can be transferred from the high-resolution cars-196 dataset to the low-resolution Boxcar's dataset. Several neural networks will be trained on the cars-196 dataset using different training strategies proposed in [22] and then fine-tuned on the BoxCars dataset. Multitask and auxiliary learning are then explored to measure their effect on fine-grained classification. Multitask learning, seen in [20], [21], classifies the individual parts of the fine-grain label. Auxiliary learning [23] will build on the multitask

learning models and use the separate labels as extra tasks to act as regularisation for the main task, a single fine-grained label. Finally, a combination of knowledge transfer and auxiliary learning will be experimented to investigate if they can boost the performance further when combined. Channel pooling [24] was found to work well in high-resolution vehicle datasets. This paper will investigate if the same occurs on low-resolution data.

A. Baseline architecture

As baseline deep network, the Vgg16 model will be used. Paper [16] found that the Vgg16 model gives the best results for the boxcars dataset. The final layer is replaced with a linear layer with the correct number of classes to classify a single fine-grain label, i.e. the concatenation of make, model, submodel and year/generation. Additional layers are added to implement multitask and auxiliary learning models which are explained in Multitask Learning and Auxiliary Learning subsections in the Technical Section. The performance of each experiment will be evaluated on the testing split of the BoxCars dataset.

B. Fine-to-Coarse Knowledge Transfer

Paper [22] proposed that knowledge learnt from a high-res domain can be used to improve classification on low-res domain images. They proposed a novel staged training strategy to train a deep neural network. They trained the AlexNet network on the high-resolution cars-196 dataset. The network was then fine-tuned on a downsampled version of cars-196 to mimic a low-resolution dataset. They were able to show that networks trained, using the staged training strategy, performed better when evaluated on the low-resolution dataset compared to training a model on the downsampled dataset alone. This paper validates the approach of transferring knowledge between different resolutions of the cars-196 dataset. This paper builds on this research and proposes the approach of transferring knowledge between the different resolution of the cars-196 dataset to the boxcars dataset to improve classification on low-resolution data. Various models will be trained using the training strategies outline in [22] and then fine-tuned on the boxcars dataset to investigate if the training strategies affect how knowledge is transferred. Paper [22] used the AlexNet model, while this paper will use the Vgg16 model. The Vgg16 model is more complex and has shown to perform better than AlexNet in the ImageNet competitions, so a better result is expected**.**

This experiment will follow the same training process from paper [22] to create various models trained on the different versions of the cars-196 before fine-tuning them on boxcars dataset. A Vgg16 model will be initialised with pre-trained weights from ImageNet before training on one of four training strategies taken from paper [22]. Each model will be trained using the categorical cross-entropy loss function**.** The high training strategy (TS) will train the model during a high-resolution version of cars-196. The low TS trains the model using the low-resolution version of cars-196. This version of cars-196 is manually created by taking the high resolution images and downsampling it to the boxcar's resolution. The boxcars dataset uses multiple cameras with different viewpoints, so the resolution varies. The average resolution of boxcars is 122 x 122. The high-resolution cars-196 dataset is downsampled from 700 x 480 to this, to match the boxcars dataset. Examples of this can be seen in Figure 1. The Staged TS trains a model on the high-resolution cars-196, then fine-tunes it on the low-resolution dataset. The Mixed TS trains a model on the high resolution and low-resolution cars-196 dataset simultaneously. All samples in all strategies are then re-scaled to 224 x 224 to match the input resolution required by Vgg16.

Figure 1 - Example of the high-resolution (top) and low resolution (bottom) cars-196 vehicle images. The low resolution version is downsampled to the resolution of the boxcars dataset (122 x 122)

C. Multitask Learning

Multitask learning is the concept of learning several outputs from a single image simultaneously. In the case of fine-grain vehicle classification, a model trained, using the multitask learning approach, is able to predict multiple labels independently for a single image. Each boxcar image has a single fine-grained label to describe the vehicle. These labels can be split up into separate labels, to train a multitask learning network. The architecture and loss function to perform multitask learning needs to be modified from the previous section to output a prediction for each task and train the model considering each task.

1) Loss Function

Multitask learning outputs multiple tasks at the same time that need to be taken into account when the loss function is calculated, and the error backpropagated. A conventional loss function used to train a deep learning network to classify images is categorical cross-entropy. This is used to compare the distributions of the predictions with the true distributions. An equation for this can be seen in Error! Reference source not found.:

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * \log(\hat{y})) \qquad (1)$$

where y is the true value, $\hat{y}$ is the predicted value, M is the number of classes and N is the batch size.

When the models are fine-tuned on the Boxcar's dataset in Fine-to-Coarse Knowledge Transfer subsection, there are 109 different outputs (109 unique fine grain labels) in the final layer. When the model receives an image during training, a probability for each class is output from the final layers. This is compared against the true value using the categorical cross-entropy equation to calculate the loss. During multitask

learning, there are multiple output layers, each with their own set of classes. The boxcars dataset contains 16 makes, 68 models, 6 submodels and 7 generations. Each output needs to be taken into account when calculating the loss to ensure all tasks affect the model during training. The categorical loss function is used to calculate the loss for each output layer in this instance. The losses for each task are then combined to calculate a total loss. The weight of each task on the overall loss can be used to model different relationships between the tasks. For fine-grain labels, the labels follow a hierarchical structure with higher and lower level parts. By multiplying each task by a different lambda, multiple relationships can be modelled. The equation to calculate the loss when the boxcars multitask model is trained is shown in equation 2:

$$loss_{boxcar} = (loss_{make} * \lambda_{make}) + (loss_{model} * \lambda_{model}) + (loss_{submodel} * \lambda_{submodel}) + (loss_{generation} * \lambda\_generation) \quad (2)$$

Each loss is calculated using the categorical cross entropy loss function. Each loss is multiplied by a separate λ.

2) *Architecture*

The final layers in a neural network are used to output the model's prediction for each class. In the Fine-to-Coarse Knowledge Transfer subsection, the output task was a single fine-grained label for each image. To accomplish this, the final layer was modified to contain the number of outputs depending on the dataset used. (Figure 2)

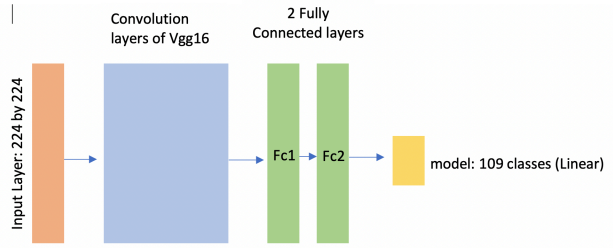

Figure 2 - Neural network architecture for a model that returns a single output for each input. This network has 109 outputs to match the number of unique fine-grain labels in the boxcar's dataset. Each fully connected layer consists of a linear layer with an input and output of 4096 features followed by a dropout layer with a probability of 0.5.

To accomplish multitask learning, an output layer for each task is necessary. For the Boxcar's dataset, four output layers are added, one for make, model, submodel and generation. An example of this architecture can be seen in Figure 3. The way data is transferred through a multitask neural network to each output layer can affect the performance of each task. The input to each final layer in Figure 3 is the output from Fc2. This transfer of data gives the impression that each task is equally important because they are all using the same feature map to calculate their class predictions. As an alternative, fine-grain vehicle labels follow a hierarchical structure that can be modelled by transferring the data differently. These changes could boost the performance of each task. Figure 4 shows two more model configurations that change how the data is transferred to each output layer. Model B uses the fact that there is a hierarchical relationship between the labels. The higher-level feature maps are passed into the lower levels along with the feature map from fc2. The cascading information passed to each layer could improve the accuracy of the lower-level tasks. This is motivated by the fact that there is a relationship between the labels that follow a hierarchical structure. Model C looks at the hierarchical structure exclusively and attempts to see if passing information from the previous task can lead to a less complicated network that can still perform well.

As the final prediction in all strategies, the fine-grain label is reconstructed by concatenating the output label of each task. This allows it to be compared to other approaches.

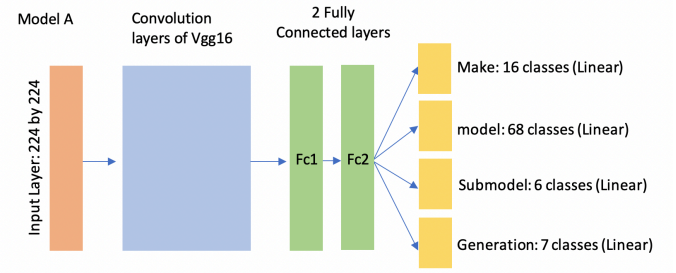

Figure 3 - Multitask network architecture for the Boxcars dataset. This will be known as Model A. Each task receives the same input from the fc2 layer. The output layers are all linear layers with the number of output equal to the number of unique labels for that task

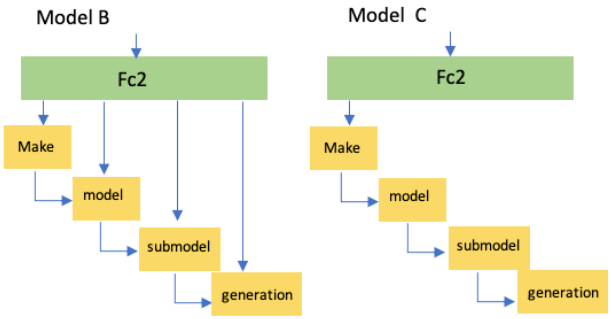

Figure 4 - Final layers for multitask learning for Model B and C. The fc2 layer is continued from the fc2 layer in Figure 3. Model B takes the input from the fc2 layer and uses this as input to each output layer. The predicted label from the higher-level tasks is input into the lower-level output layers. Model C passes the output from the fc2 layer to the make output layer. The predictions from each task are then passed to the next task.

D. *Auxiliary Learning*

Auxiliary learning is the process of learning extra tasks alongside the main task to help boost the performance. These additional tasks act as regularisations to the main task to help create a more robust model. [23] shows that the addition of auxiliary tasks in a multitask neural network boosts the performance and decreases training times. Conventionally, auxiliary tasks are tasks that may not directly affect the main task but can help the model understand more about the input. If a neural network is trained to annotate its surroundings while driving, the weather or lighting conditions may not directly help to interpret the surroundings but could act as auxiliary tasks to help understand the environment more, creating a more robust model. This paper will use the network architecture from the Fine-to-Coarse Knowledge Transfer

subsection to predict the main task. The multitask learning networks will act as auxiliary tasks. These two networks are combined to make the auxiliary learning networks. The multitask architecture that receives the best results will be used to design the architecture for the auxiliary tasks. An example of the auxiliary network can be seen in Figure 5.

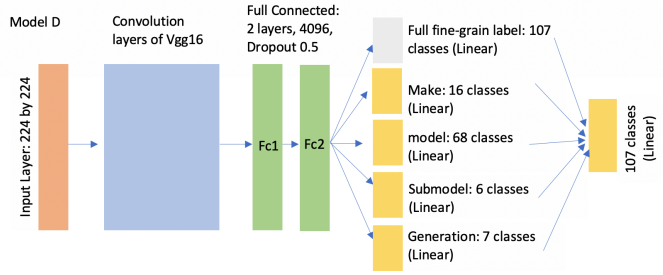

Figure 5 - Auxiliary network architecture for the Boxcars dataset. This will be known as Model D. The auxiliary tasks pass data similar to model A from the multitask learning section. Each task calculates a prediction, including the main task. These are all passed into the final layer to re-predict the full fine-grain label. All yellow boxes give an output when an input is passed through the model. There are five outputs for each auxiliary model.

Additional network architectures are proposed to investigate how passing the feature maps in the final layers affects the performance of the auxiliary networks. Model D in Figure 5 works on each task separately and then combines them to predict the final label. Model E in Figure 6 follows the classic implementation of auxiliary learning, where the auxiliary tasks are learnt alongside the main task and used as extra information to feedback into the model to predict the final label. Finally, model F in Figure 6 uses the auxiliary tasks by themselves to predict the fine-grained label. The auxiliary tasks in each network use the output from the fc2 layer to make their prediction which is similar to how it is done in model A.

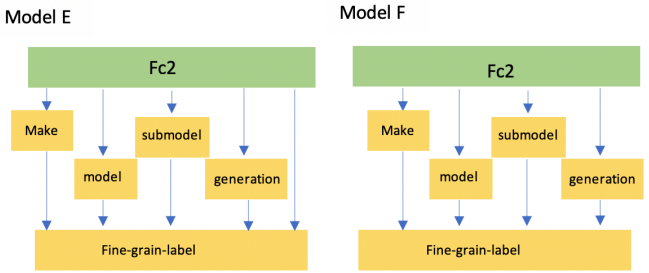

Figure 6 – Each new auxiliary network is continued on from the fc2 layer in Figure 5. Model E uses the predictions from the auxiliary tasks and the fc2 output to calculate the full fine-grain label. Model F uses the auxiliary task only to output the final label. The dataflow to each auxiliary task model that of model A (Figure 3)

Similar to the Multitask Learning subsection, a loss function and several network architectures are proposed to investigate the approach. The full fine-grain label, along with the individual parts of this label, are all output from the auxiliary network, so a new equation is required to ensure all the tasks are taken into account. This equation is proposed in equation (3). Each output layer of the network calculates their loss using the categorical cross-entropy equation (1). Each task's loss function is multiplied by different lambdas to enable each task to be weighted differently during training.

$$loss_{boxcar} = (loss_{main} * \lambda_{main}) + (loss_{make} * \lambda_{make}) + (loss_{model} + \lambda_{model}) + (loss_{submodel} * \lambda_{submodel}) + (loss_{generation} * \lambda_{generation}) \quad (3)$$

Each loss is calculated using the categorical cross entropy loss function. Each loss is multiplied by a separate lambda.

E. Channel Max Pooling

[17] prosed a new layer called Channel Max Pooling (CMP). The proposed CMP layer divides the input feature maps into several sub-groups. Then it compresses the feature maps within each sub-group into a new one. (Figure 7) The compression is done by selecting the maximum value at the same location in each feature maps within the sub-group. This can lead to more discriminative features being found due to the smaller number of feature maps. This, in turn, can lead to a better generalised CNN. The layer is added between the last max-pooling layer and the first fully connected layer. This will modify the feature map passed into the fully connected layers. This paper will validate if the addition of a CMP layer to a neural network boosts the performance on the low-resolution dataset, Boxcars.

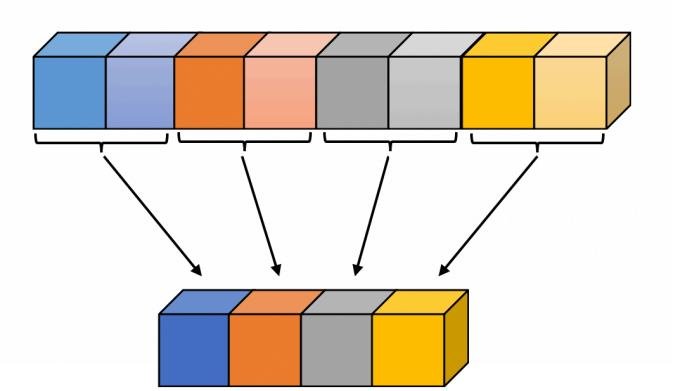

Figure 7 - Diagram of how the CMP layer transforms a feature map when a compression factor of 2 is used. The input is split into subgroups (2 cubes). Each subgroup finds the maximum value at each location across the subgroup channels. The output from each subgroup is concatenated to give an output that reduced the number of channels by 2(compression factor) but retained the width and height of the input feature map.

IV. Results and Analysis

The Vgg16 neural network will be used in all experiments due to the research shown in [16]. The Vgg16 model was shown to work best on the Boxcars dataset compared with other state-of-the-art neural networks. This is the dataset that will be used to evaluate the performance of fine-grain classification in this paper. The neural network will be initialised with ImageNet weights to speed up training. The network will be trained using a constant learning rate of 1e-4 and the Adam optimisers with default parameters. The boxcars paper used the SGD optimiser. When the Adam optimiser is used to train the base architecture on the boxcar's dataset, a much better performance was obtained. The results are shown in the

appendix. All networks trained throughout this paper will use the Adam optimiser. The Vgg16 model will be used as a base and modified depending on the experiment. These modifications can be seen in the corresponding Technical Section subsections.

A. Properties of Fine-Grain Datasets

The Related work section outlined several large-scale datasets that can be used to perform vehicle classification. This paper will look at the boxcars and cars-196 dataset since they contain samples with a variety of poses in fine-grained detail and have enough images to train a neural network.

1) BoxCars Dataset

BoxCars datasets [13] contains surveillance camera images which can be used to perform vehicle classification on realistic data. The dataset includes 116,286 vehicle images taken from multiple cameras under different viewpoints, illuminations, resolutions and occlusions. The dataset provides information about the 3D bounding box for each vehicle and an image with a foreground mask extracted by background subtraction, but this will not be used in this paper. The dataset has been split up into multiple sections to allow different detail of granularity to be tested. This paper will use the hard split which contains 93,603 images. Each image is labelled with a fine-grained label with information on the make, model, submodel and generation. This section of the dataset contains 107 unique fine-grained classes. This can be split up into 16 makes, 68 models, 6 submodels and 7 generations. The generation refers to the version of that specific car opposed to the year it was built.

This dataset will be used during all experiments in this paper. The 107 fine-grained classes will be used in during the Fine-to-Coarse Knowledge Transfer section when the loss function in equation (1) is used. The fine-grained labels will be split up into individual labels for make, model submodel and generation when equation (2) and **Error! Reference source not found.** are used. The dataset outlines its' own method to split the data into fixed training and testing splits. The training split will contain 55%, the testing will contain 42% while the validation will contain 2% of the boxcar dataset. These fixed splits are used in the paper to train and evaluate each model.

2) Cars-196 Dataset

Cars-196 [5] contains fine-grain web-natured images. This dataset contains 16,185 images. There are 196 fine-grain classes providing information about the make, model and year of each vehicle. The dataset also includes boundary boxes for each image in the dataset but again this will not be used. This dataset will be used in the Fine-to-Coarse Knowledge Transfer section as the high-resolution dataset and will be downsampled to create a low-resolution version. The dataset is already split into a training and testing split. The training split has 8144 images while the testing split has 8041 with each class split roughly 50-50 between splits. These are fixed splits to ensure that the same data is used during all experiments, and there is no crossover.

B. Fine-to-Coarse Knowledge Transfer

The first part of this section will train a neural network using the cars-196 dataset with the various TS outlined in the Fine-to-Coarse Knowledge Transfer section above. The baseline network will be used, with the output layer modified from 1000 dimensions to 196. (Figure 2) This is the number of categories in the Cars-196 dataset. During training, the model will use the loss function in equation (1). Table 1 summarises the result proposed in [22] for each TS using the updated network. Each model is trained and evaluated using the same training and test splits. The results in Table 1 are found when the model is evaluated on the high and low version of the cars-196 dataset separately. The same images are used in both test splits but are downsampled in the Low cars-196 version.

Training Strategy	Evaluated on High Cars-196 (%)	Evaluated on Low Cars-196 (%)
High	79.7	66.3
Low	75.4	75.7
Staged	77.2	73.1
Mixed	73.7	70.3

Table 1 - Networks trained using different training strategies and evaluated on the high and low-resolution cars-196 dataset separately.

Contrary to the results seen in [22], the performance of the network trained on the Staged TS is worse than the network trained on the low TS when evaluated on the low resolution version of cars-196. This is brought about by two factors. The cars-196 low-resolution version is not downsampled as much in this paper compared to [22]. The paper modified the images to have a resolution of 50 x 50, whereas this paper transforms them to be 122 x 122 to match that of the Boxcar's dataset. The downsampled images in this paper are high quality so more features can be picked out, leading to better performance. The other factor is that the Vgg16 is a more complex network than AlexNet. It was shown to be much better at image classification during the ImageNet competitions. Vgg16 has more layers with filters that convolute 3 x 3. This high number of filters and small kernel size allows more discriminate features to be learnt at each layer, giving a better result. The knowledge learnt during the staged TS adds unnecessary noise, causing the performance on the low resolution to suffer**.** One interesting observation is that the network is still able to perform relatively well when trained on the low-resolution dataset and evaluated on the high-resolution dataset. The mixed dataset performs worse when evaluated on the high dataset and better on the low dataset**.** This is expected because the model is trying to fit itself to both datasets. The model will be able to distinguish between the low-resolution images better, but this will cause more noise affecting the results on the high-resolution images.

The models trained in Table 1 are fine-tuned on the BoxCars dataset giving the results seen in Table 2. The output layer of each neural network was modified to have 107 outputs to match that of the Boxcar's dataset while the rest of the layers have the same weights as the previous model. The training and testing split outlined in BoxCars Dataset is used to train and evaluate each model in this section. In Table 2, the BoxCars TS shows the baseline result received when the Vgg16 model is trained on the BoxCars dataset alone. The rest

of the models are fine-tuned on a model from Table 1. It is observed that when a network is trained using a TS that includes the high-resolution cars-196 dataset, a worse testing accuracy is achieved. An improvement of 0.7% can be seen when fine-tuning from the low TS model. This result shows that fine-tuning a model on one dataset with the same resolution can be transferred over to another dataset even if it is a small improvement.

Training Strategy	Test on BC (%)
BoxCars	84.1
High + BoxCars	82.2
Low + BoxCars	84.8
Staged + BoxCars	81.8
Mixed + Boxcars	83.1

Table 2 - Networks trained on various training strategies and evaluated on the Boxcars dataset

The investigation above indicated that knowledge can be transferred between datasets when they are of a similar resolution. A furthered experiment is conducted to evaluate how the number of samples used to fine-tune a model affects performance. The low TS model is used as a base, and several models will be fine-tuned with different percentages of the boxcars training split. The models will be evaluated using the whole testing split discussed in the BoxCars Dataset subsection. The result of this can be seen in Figure 8. This graph also includes the results when a Vgg16 model with 109 outputs in its final layer, is trained with ImageNet weights.

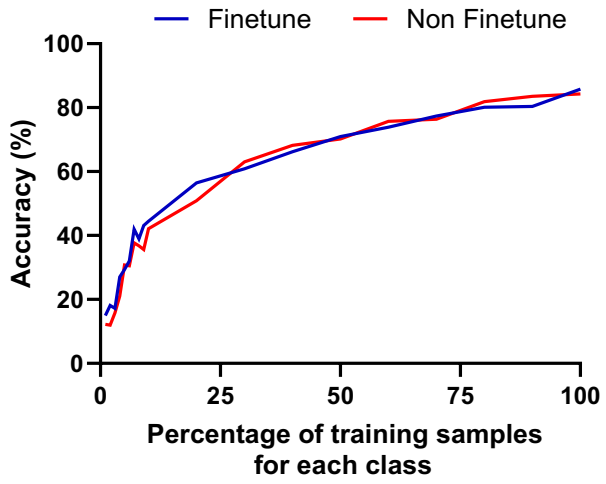

Figure 8 - Testing accuracy received when a network is fine-tuned from the low TS network on a decrease training split and evaluated on the testing split of Boxcars dataset. (fine-tune) The non-fine-tune results outlines the accuracy obtained when the network is initiated with ImageNet weights.

The difference between the fine-tuned model and non-fine-tuned model is very small. The fine-tuned model gave a slightly better accuracy up until 40% of the training split. After this, both model performances are similar. It can be seen that the best accuracy is achieved when the whole dataset is used. This would indicate that if more data was added to the dataset, further improvement could be made. This proves that fine-tuning on a similar resolution dataset does help when a smaller dataset is used to fine-tune the model, but the overall effect is minimal.

C. Multitask Learning

This section evaluates the performance of the multitask learning network outlined in Figure 3 and Figure 4 on the boxcars dataset. The networks are trained and evaluated using the splits described in the BoxCars Dataset section. The loss function is calculated using equation (2). Experiments were run to investigate how each network architecture and lambdas configuration affects the individual task and reconstructed label performance. Table 3 shows the individual task accuracies obtained when the loss function used to train each network uses different lambda configurations. The lambda configurations are outlined in the figure. The final column contains the accuracy of the reconstructed label. This is found by taking the best prediction from each task and concatenating them together. This is then compared against the full fine-grain label. A graph of the constructed accuracies for each network and lambda configuration is shown in Figure 9.

Model	Make	Model	Submodel	Generation	Reconstructed Label
A	96.4	92.1	95.0	92.7	82.9
B	93.4	89.6	93.1	92.2	83.3
C	92.0	87.7	92.8	91.0	81.3
A	93.4	88.8	93.0	91.7	83.7
B	93.0	89.1	93.8	91.9	83.8
C	92.6	87.9	92.4	90.6	80.9
A	92.5	88.5	93.7	91.2	82.4
B	92.2	88.2	93.1	91.4	82.5
C	91.8	87.6	93.3	91.0	81.7

Table 3 - Performance of each task in each multitask learning network against the boxcars testing split. The top table uses lambda options 1. All tasks in this option are multiplied by 0.2. The middle table uses lambda option two were the make is multiplied by 0.8, model by 0.6, submodel by 0.4 and generation by 0.2. The bottom table uses option 3 were the make is multiplied 0.2, model by 0.4, submodel by 0.6 and generation by 0.8.

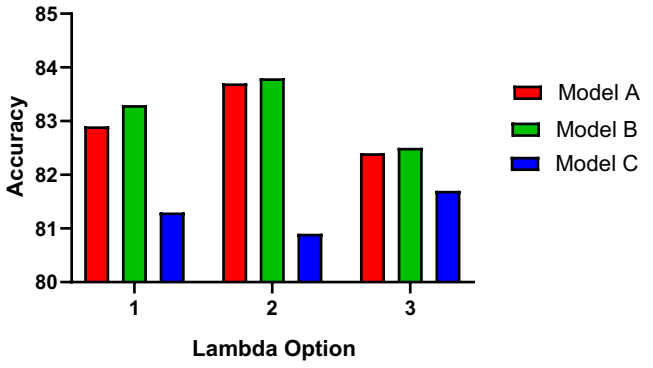

Figure 9 - Reconstructed accuracy of each multitask network architecture and lambda configurations.

It is observed that different architectural structures and lambda configurations affect the performance of the individual tasks and reconstructed labels when evaluated on the boxcar's dataset. Lambda option one models that all tasks are equal. This is the same relationship modelled in model A. Lambda option two models a hierarchical relationship between the higher-level tasks and low-level tasks. The order goes from

make, to model, to submodel, to generation. This relationship is also modelled in model B and C. Lambda option three models the opposite of the hierarchical relationship to see how it affects the performance.

Across all lambda configurations in Figure 9, model B performs best, followed by model A. The two best-performing networks used lambda option two during training, indicating that modelling a hierarchical relationship between the tasks is important. This is proven multiple times in Figure 9. Model B's architecture models the same hierarchical relationship that option two but in its architecture between the tasks. Across each lambda option, this network architecture performs best. When training model B using lambda option 2, the best performing network is found. Another observation is that the worst-performing A and B models are observed when the inverse hierarchical relationship is modelled (option three). This shows that when the inverse relationship is used, it hinders the performance. These networks perform worse when the same networks are trained using a loss function that models each task equally (option one)

Model C was used to investigate if there was enough information passed from the higher-level tasks to predict the lower-level tasks. It is observed that there is enough information to do this, but the reconstructed accuracy is decreased significantly compared to model B, which use this information as extra data for each task. When different lambdas configurations are used with model C, the opposite relationship is observed compared to model A and B. Model C performs best when trained using option three, then one then two. This would suggest that with the less complicated data flow in the final layer, that only follows the hierarchical structure, weighing the lower-level tasks higher in the loss function compared to the higher ones performs better.

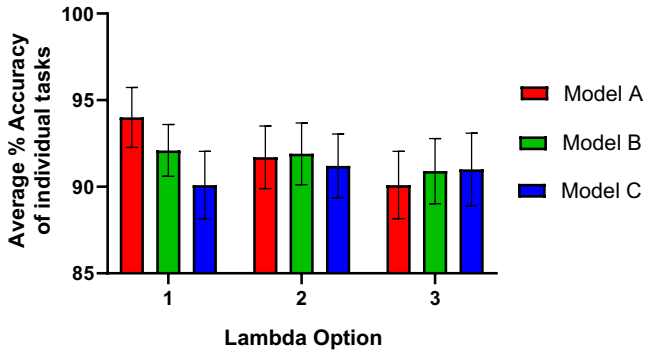

Figure 10 - Average accuracy across tasks of each multitask network architecture and lambda configurations.

An interesting observation can be seen in Figure 10. This graph shows the average across the individual tasks for each network and lambda configuration. This graph would give the impression that model A trained with lambda option one, would perform best when the reconstructed accuracy is calculated because it received the highest average accuracy. From the previous experiments, this is not the case. Model A, option 1 models an equal relationship between the tasks. During training, model A's goal, with its architecture and lambda option, is to improve all task equally. This network isn't trained in a way to understand that the labels are linked. When the reconstruction occurs, the best prediction from each task is used. If one of these is wrong, then the whole label is wrong. With networks that train using an aspect of the hierarchical relationship similar to that seen in the labels, the reconstructed labels were better. This is because the networks have learnt that a low-level task is a specific label because of the higher-level predictions.

Overall, this experiment proves that modelling the hierarchical relationship between the tasks can be leveraged to boost the performance of the network on the reconstructed fine-grained labels. Modelling a hierarchical relationship in the loss function has been shown to increase the performance more than modelling it in the output layers of the network. Both can still be utilised to boost the performance together. Comparing the best results from this section to the baseline in Table 2 indicates that multitask networks are not able to perform as well as a single output layer. There is a 0.3% decrease in performance compared to the baseline network.

D. Auxiliary Learning

The previous section showed that a multitask learning network performed slightly worse than the baseline network set out in Table 2. It was shown that networks that model a hierarchical relationship, which fine-grain labels follow, perform best. This section will take the best performing multitask architectures and use these to create the auxiliary tasks for three auxiliary learning networks outlined in Figure 5 and Figure 6. These networks already model the dataflow in the final layers seen in model A. The same set of networks were created to represent model's B data flow in the final layers. This new version of D, E, and F will be called G, H, and I. This means there are two versions of the same auxiliary network evaluated in this section, but they differ in how the data is transferred across the auxiliary tasks. Each network is trained using the loss function outlined in equation (3). This equation includes the loss from the main task, a single fine-grain label. The top-performing network from the previous section used lambda option one and two during training. These lambda options are used again with the main loss multiplied by 1 to ensure the main task is learnt over the auxiliary tasks. Each network is trained and evaluated using the boxcars dataset. Figure 11 shows the results of each variation of auxiliary network, architecture structure and lambda option.

The first observation made from the graphs in Figure 11 is that all the auxiliary learning networks perform better than the baseline from Table 2.(84.1%) The lowest-performing auxiliary network receives an accuracy of 84.2% whilst the highest performing network obtained an accuracy of 85.3%. The best performing networks occur when model D is trained on lambda option one or Model E is trained on lambda option 2. Figure 11a) indicates that it is better to model an equal relationship between tasks in the loss function compared to a hierarchical one. Model D and G perform better when trained using option 1. The opposite observation is seen in Figure 11b). These networks show that a better performance is found when the hierarchical relationship is modelled in the loss function compared to the equal relationship. In graph a) and b), using the best lambda option for both, the best network is found when the auxiliary tasks model the inverse relationship

from the loss function used. Model G, which models the hierarchical relationship in the architecture, works best when trained on a loss function which models that each task is equal. This is the opposite for model E.

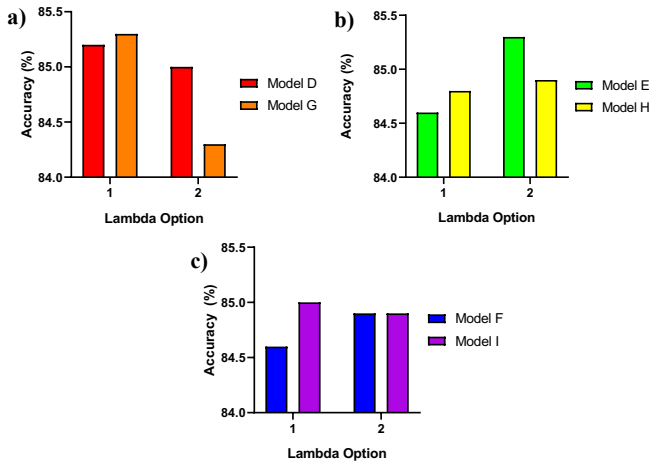

Figure 11 – Fine-grain accuracy obtained by each auxiliary network when different lambda options are used. Graph A uses the architecture were each task, including the main task provides a predicating. These predictions are then passed into a final layer to get the full fine-grain label prediction. Graph B uses the prediction from each auxiliary task and the feature map from fc2 to calculate the fine fine-grain label. Graph C networks use the predictions from the auxiliary tasks as input to the final layer to predict the full fine-grain label

Figure 11c) gives a similar result across each auxiliary network version except with model F, option one, where there is a drop in performance. This architecture was designed to investigate if the addition of an output layer to the end of the multitask networks would give better accuracy than reconstructing the full fine-grain label. This proves that learning the full fine-grain layer as an output at the end of the network performs better than reconstructing the full label in the previous section. The best network, in this case, is Model I, option one where the hierarchical relationship is modelled in the auxiliary task data flow rather than the loss function. This was not the best network from the previous section but only performed 0.2% better than the top two networks from multitask learning.

The main take away from this section is that the use of different auxiliary networks leads to a very complex network that can be leveraged to boost the performance of fine-grain classification. These networks need to be designed and trained with specific conditions to ensure that they give the best performance they can. The two networks that performed best in this section are representative of that. They both need different loss functions and architectures to get the same results. Modelling the hierarchical relationship in the loss function and architecture is too complex for these models, so using one is better. Figure 11a) networks perform better using a simpler loss function with the hierarchical relationship in the architecture. In contrast, Figure 11b) networks perform better when there is a simpler architecture, and the hierarchical relationship is in the loss function.

All auxiliary networks output the auxiliary tasks along with the main tasks to allow the overall loss to be calculated. The graph in Figure 12 outlines the average accuracy between the individual tasks for each network and lambda configuration. When compared against the same multitask network results in Figure 10, the standard deviation calculated is lower for the auxiliary network. This would indicate that the tasks work better than the multitask network to create a more robust model.

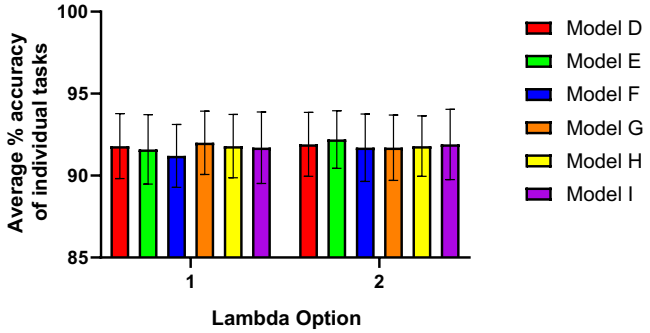

Figure 12 - Average accuracy across tasks of each auxiliary network architecture and lambda configurations.

E. Auxiliary Learning and Fine-to-Coarse Knowledge Transfer

The results from the Fine-to-Coarse Knowledge Transfer section shows that a small amount of knowledge can be transferred when similar domain datasets are used to train a neural network. The auxiliary learning networks also displayed a boost in performance with two networks able to improve on the baseline by 1.2%. Combining both these techniques could lead to an even better performing network overall. To accomplish a fine-tuned auxiliary model, the low TS network is used as a base because it gives the best result when fine-tuned on the Boxcars dataset. (Table 2) To convert the base to an auxiliary network, the final layer was removed, and the best performing final layers in the auxiliary network were added. In this case, Model G and E. Lambda option one will be used to train model G and option two to train model E using equation (3). These configurations were able to find the best performing networks in the auxiliary learning section. Each network was then fine-tuned using the training split of the boxcar's dataset. Table 4 includes the performance of the baseline and low to Boxcars results from Table 2, the performance of model G, option one and model E, option two and the new fine-tuned auxiliary network performances. The performance for each network when evaluated on the boxcar's dataset is shown.

Model	Tested on BC (%)
Baseline	84.1
Low + BC	84.8
Model G, Option 1	85.3
Model E, Option 2	85.3
Fine-tune Model G, Option 1	86.1
Fine-tune Model E, Option 2	85.1

Table 4 – Accuracy obtained when the best auxiliary networks are fine-tuned from the low resolution version of cars-196. The table also includes the results from previous sections to compare the fine-tuned auxiliary performances.

The model G fine-tuned auxiliary network achieved an accuracy score of 86.1% when evaluated on the boxcar's dataset. This is an increase in accuracy of 0.8% between the auxiliary and fine-tuned auxiliary network. This increase shows that knowledge is transferred from the low-resolution cars-196 dataset to the boxcar's dataset. The fine-tune model E performs worse, indicating that the knowledge transfer hinders its' performance. One key difference between model G and E is that G learns a representation of the fine-grain label along with the auxiliary tasks. All this information is passed into the final output to calculate the same fine-grain label but with more information to base the prediction on. This additional step is not present in E and would indicate that it affects the performance when fine-tuned from a pre-trained network. The loss function used in model G is also important. When option two is used to train model G, an accuracy of 85.1% is obtained. This shows that a specific architecture and loss function is needed to boost the performance.

F. Validating Channel Pooling on Low-resolution dataset

Paper [17] achieved a 0.1% increase in accuracy when the Vgg16 model was trained and evaluated on the cars-196 dataset. This section investigates if the addition of CMP can boost the performance of a neural network trained on a low-resolution dataset. The network outlined in Figure 2 will be modified to contain the CMP layer between the last max pooling and first fully connected layer. This addition will change the dimensions of the pooled features from 512x7x7 to cx7x7. c is calculated by taking the number of channels from the last max-pooling layer (512) and dividing it by the compression factor r. This is a new hyperparameter proposed by the paper and can significantly affect the information passed to the fully connected layer. The paper used the compression factor 2,4,8,16 and 32 during training to evaluate how the models deal with the different pooled features. It was found that a compression factor of 2 achieved the best result for the Vgg16 model.

The results from this experiment are found in Table 5. The Vgg16 performance is found from the baseline result in Table 2. The addition of the CMP shows an improvement over the baseline of 0.4%. This result was found when the compression factor was set to 2. The time taken to train the model with a CMP layer took 13.5 hours. It only takes 2 hours to train the same network without the CMP layer. The extra time occurs because the method used to implement CMP doesn't fully utilise the GPU accelerations that PyTorch offers. When a more efficient implementation is created, further investigations on how different compression factors affect the performance of Vgg16 on the low-resolution images will be conducted.

Network	Test on Boxcars (%)
Vgg16	84.1
Vgg16 + CMP	84.5

Table 5 - Accuracy obtained by the baseline and modified networks trained and evaluated on the boxcar's dataset.

V. Conclusions

In this paper, the fine-grained classification of low-resolution vehicles has been studied. A boost in performance from the baseline set by [16] has been improved from 77.3% to 84.1% by using the Adam optimiser over SDG. Using this updated baseline, it has been proven that the addition of transfer learning and auxiliary learning can be used to improve the classification further. Transfer learning from a dataset of a similar resolution to the boxcar's dataset boosted the performance by 0.7%. The auxiliary learning approach increased from the baseline by 1.1%. When the approaches were combined, there was a boost of 2% in performance. This shows each approach offers different improvements to the network that can stack. Multitask learning obtained a worse performance when the reconstructed labels were used to calculate the accuracy. This approach didn't perform as well as the baseline but offered useful insight into how the architectures and loss function configurations affect each task. This allowed auxiliary networks to be created, modelling the best multitask configurations, leading to better auxiliary learning networks. Finally, the channel pooling investigation obtained a small improvement when trained and evaluated on the boxcar dataset. Further investigations will occur when the training time for networks with CMP layers decreases to observe how the Vgg16 network and different compression factors perform on the low-resolution dataset.

Further research should look at boxcars multitask networks fine-tuned on the low-resolution version of the cars-196 dataset. This could boost the performance similar to the fine-tuned auxiliary networks. The auxiliary network could then be fine-tuned on the fine-tuned multitask networks to see if any extra information is passed between networks.

Appendix

The paper [16] used the SGD optimiser during training to the modified the Vgg16 to classify vehicles from the boxcars datasets. When the Adam optimiser is used to train the same network, a significant boost in the performance is found. These results are seen in Table 6. This optimiser is used when training all networks in this paper to ensure the best performance if obtained with each approach.

Optimiser	Test on Boxcars (%)
SGD	77.3
Adam	84.1

Table 6 – Accuracy obtained when networks trained on the training split of the boxcar's dataset using various optimiser are evaluated on the testing split of boxcars

Acknowledgement

Thanks to Jesus Martinez del Rincon for the guidance and support throughout the year as supervisor. Thanks also is due to the members of the PyTorch forum. They were an invaluable resource during this research project

References

[1] G. Pearce and N. Pears, "Automatic make and model recognition from frontal images of cars," in *2011 8th*

IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2011, pp. 373–378.

[2] D. F. Llorca, D. Colás, I. G. Daza, I. Parra, and M. A. Sotelo, "Vehicle model recognition using geometry and appearance of car emblems from rear view images," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 3094–3099.

[3] L. Liao, R. Hu, J. Xiao, Q. Wang, J. Xiao, and J. Chen, "Exploiting effects of parts in fine-grained categorization of vehicles," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 745–749.

[4] J. Hsieh, L. Chen, and D. Chen, "Symmetrical SURF and Its Applications to Vehicle Detection and Vehicle Make and Model Recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 6–20, 2014.

[5] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 554–561.

[6] J. Deng, J. Krause, and L. Fei-Fei, "Fine-Grained Crowdsourcing for Fine-Grained Recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 580–587.

[7] A. Schumann, L. Sommer, K. Valev, and J. Beyerer, "A systematic evaluation of recent deep learning architectures for fine-grained vehicle classification," in *CoRR*, 2018, vol. abs/1806.0, p. 1.

[8] D. Liu and Y. Wang, "Monza: Image Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning," 2016.

[9] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3973–3981.

[10] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," in *2016 IEEE International Conference on Multimedia and Expo (ICME)*, 2016, pp. 1–6.

[11] Z. Tang *et al.*, "CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification." 2019.

[12] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford Campus vision and lidar data set," *Int. J. Rob. Res.*, vol. 30, no. 13, pp. 1543–1552, Mar. 2011.

[13] J. Sochor, A. Herout, and J. Havel, "BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 3006–3015.

[14] D. Cai, K. Chen, Y. Qian, and J.-K. Kämäräinen, "Convolutional Low-Resolution Fine-Grained Classification," *CoRR*, vol. abs/1703.0, 2017.

[15] M. Dubska, S. Jakub, and A. Herout, "Automatic camera calibration for traffic understanding," *BMVC*, 2014.

[16] J. Sochor, J. Špaňhel, and A. Herout, "BoxCars: Improving Fine-Grained Recognition of Vehicles Using 3-D Bounding Boxes in Traffic Surveillance," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 97–108, 2019.

[17] Z. Ma *et al.*, "Fine-Grained Vehicle Classification with Channel Max Pooling Modified CNNs," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3224–3233, 2019.

[18] Q. Hu, H. Wang, T. Li, and C. Shen, "Deep CNNs with Spatially Weighted Pooling for Fine-Grained Car Recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3147–3156, 2017.

[19] X. Li, L. Yu, D. Chang, Z. Ma, and J. Cao, "Dual Cross-Entropy Loss for Small-Sample Fine-Grained Vehicle Classification," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4204–4212, 2019.

[20] B. Solmaz, E. Gundogdu, V. Yucesoy, A. Koç, and A. A. Alatan, "Fine-grained recognition of maritime vessels and land vehicles by deep feature embedding," *IET Comput. Vis.*, vol. 12, no. 8, pp. 1121–1132, 2018.

[21] Y. Tian, D. Zhang, C. Jing, D. Chu, and L. Yang, "Multi-task convolutional neural network for car attribute recognition," in *2017 4th International Conference on Systems and Informatics (ICSAI)*, 2017, pp. 459–463.

[22] X. Peng, J. Hoffman, S. X. Yu, and K. Saenko, "Fine-to-coarse Knowledge Transfer For Low-Res Image Classification," *CoRR*, vol. abs/1605.0, 2016.

[23] L. Liebel and M. Körner, "Auxiliary Tasks in Multi-task Learning," *CoRR*, vol. abs/1805.0, 2018.

[24] Q. Hu, H. Wang, T. Li, and C. Shen, "Deep CNNs With Spatially Weighted Pooling for Fine-Grained Car Recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3147–3156, 2017.