Combattre le débalancement de données dans un jeu avec une faible quantité de données

Léo Backert
Département d'informatique et de génie logiciel
Université Laval
Québec, QC G1V0A7
leo.backert.1@ulaval.ca

Abstract

Dans cette étude, nous allons nous pencher sur le problème du débalancement des données dans un contexte de détection de composantes sur les façades de bâtiments. Pour cela, nous passerons à travers trois méthodes : l'échantillonnage des données, l'augmentation des données et l'utilisation de la focal loss. Ces méthodes ont généralement permis d'augmenter la précision de nos résultats de prédictions de composantes. Cependant, cela n'est pas encore suffisant pour répondre aux exigences du projet. Pour que le projet soit réalisable, il faudrait ajouter de nouvelles données au projet, car certaines catégories ont beaucoup trop peu d'instances ce qui ne permet pas aux méthodes de résoudre le problème de débalancement des classes.

1 Introduction

Ce projet porte sur la mise en place d'un algorithme d'apprentissage machine qui a pour but d'automatisé la détection des composantes des bâtiments dans les études de fonds de prévoyance et les carnets d'entretiens proposés par Hoodi.ai.

Le carnet d'entretien a pour cible les syndicats des copropriétés de toute taille. Il a pour but d'établir un registre dans lequel les papiers administratifs sont listés (Rénovations des composantes du bâtiment, plans, factures, etc). En outre, pour l'étude de fonds de Hoodi un des facteurs marquant qui permet de réduire le coût d'une étude est la visite virtuelle (téléversement des photos par un client).

Pour ces deux produits, l'algorithme doit être capable de détecter les composantes dans une image. Il existe différents algorithmes permettant cette tâche d'après la revue de Puja Bharati et Ankita Pramanik [1]. Cependant, les données du projet sont sujettes à un fort débalancement. Cela est dû au fait que les données sont récupérées d'anciens projets. Dans la réalité on trouvera plus de certaines composantes que d'autres. C'est pour cela que cette étude se basera sur la gestion des données débalancé *premier plan – premier plan* dans un contexte de détections d'objets [2, 3].

Plus précisément, cette étude passera à travers différentes méthodes : *l'échantillonnage*, l'*augmentation de données* et l'utilisation d'une *fonction de loss personnalisé*. Appliqué à différents modèles pré-entraîner du framework de detectron2 [4].

Finalement, dans le contexte de cette étude, les méthodes d'échantillonnage et de focal loss sont les meilleures pour des modèles pré-entraîner sur COCO avec un gain d'environ 0.8 % et 0.7% de précisions moyennes sur les boîtes englobantes.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Littérature

Il existe déjà de nombreux travaux réalisés sur la détection de composantes d'habitations comme les travaux sur le OneFormer [5]. Ou encore dans la surveillance de l'état des structures [6, 7]. Ces travaux sont très proches de l'objectif du projet dans le cadre d'une étude de fond de prévoyance. Cependant, les travaux réalisés ne vont pas suffisamment profondément dans les détails des composantes. Plus précisément, ce projet ne doit pas seulement être capable de déterminer la composante (par exemple : les fenêtres), mais la gamme de composantes dans laquelle elles appartienent (fenêtres : coulissante, en PVC, ouverture en battant, etc) pour pouvoir connaître sa gamme de prix d'entretien correspondante.

Pour répondre aux exigences du projet introduit dans la section 1, le modèle `Mask R-CNN` sera utilisé [1, 8]. Car cet algorithme en plus de prédire une boîte pour la classe (bounding box) il prédit un masque. Cela peut être utile pour plusieurs raisons, la forme des classes de cette étude peuvent être discriminante sur la détection de la classe et avoir un masque permet de calculer son aire qui dans le contexte du projet peut être utile dans de travaux futurs. Plus précisément, les modèles pré-entraîner utilisé sont tiré du framework detectron2 [4]. Avec les backbones suivants : `ResNet-50` et `ResNet-100` décrit dans l'article de Kaiming He [9]. En ajoutant une structure pyramidale (`FPN`) par dessus décrit dans les modèles zoo donné par detectron2 [10].

Les données pour ce projet étant relativement limitées, on fait donc appel à une technique de transfert apprentissage (Transfer Learning). Cette méthode consiste à entraîner un modèle sur un jeu de données avec des caractéristiques similaires aux classes de notre projet afin de pouvoir réutiliser ce modèle pour adapter ses caractéristiques apprises aux classes du projet. Pour cela, on utilisera des modèles pré-entraînés sur les jeux de données COCO et Cityscapes [11, 12]. Ces jeux de données ont été choisis, car leurs images sont prises dans leurs environnements naturels et ont une grande quantité d'images extérieures. De plus, COCO contient de nombreuses classes d'éléments extérieurs dans des paysages urbains sur le continent d'Amérique du Nord, cela correspond aux données du projet qui lui contient des photos de bâtiments québécois (majoritairement des villes de Québec et de Montréal). Au contraire, le jeu de données Cityscapes contient des images urbaines de villes allemandes, mais il a des catégories plus orientées sur les composantes des bâtiments que les catégories de COCO.

Les données du projet étant prisent directement d'ancien projet de l'entreprise, elles reflètent très bien la réalité, mais amènent cependant un débalancement de données. En effet, sur une façade extérieure, on trouvera plus de fenêtres que de portes. Pour contrer cela, les revue de Justin M. Johnson et Taghi M. Khoshgoftaar et de Kemal Oksuz et al. proposent toutes deux différentes solutions pour résoudre des problèmes de débalancement *premier plan – premier plan* [2, 3]. Une des solutions les plus simple est le rééchantillonnage des données. Pour cela, on peut rajouter des données de manières à augmenter nos catégories les moins présentes dans le jeu afin d'avoir un nombre similaire d'instances de nos catégories les plus présentent. Inversement, on peut aussi retirer des données les plus présentes dans notre jeu. Dans ce projet, les données étant limitées, on va préférer en ajouter. Deuxièmement, cette méthode est très similaire à la précédentes, c'est l'augmentation des données. Cela consiste à ajouter des données déjà existantes dans le jeu original, mais en leur appliquant des transformations comme des rotations, des changements de contrastes, ... Cela peut aussi permettre au projet d'être plus robuste aux photos fournies par les clients, car leurs images ne seront pas prises avec le même angle de rotation et les paramètres de caméra ne seront pas les mêmes. Finalement, la dernière méthode qui sera considéré dans cette étude est l'utilisation d'une focal loss [13]. C'est une fonction de loss paramétrable qui permet de pénaliser plus fortement des prédictions sur des annotations des catégories les plus rare dans le jeu de données.

3 Pipeline

Pour comparer chacune des trois méthodes listé dans la section 2, nous utiliserons trois modèles différents en faisant varier l'architecture de backbone utilisé et le jeu de données de pré-entrainement. Les modèles utilisés pour la comparaison des résultats seront donc basés sur l'algorithme `Mask R-CNN` [8]. Cela permet en plus d'obtenir une boîte englobant l'objet, le modèle prédit aussi le masque binaire.

Backbones des modèles : Dans cette étude, les trois modèles comparés auront les structures des `ResNet-50+FPN` et `ResNet-101+FPN`. Dans la liste des modèles proposé par le framework detectron2 [10], il existe un modèle `ResNeXt` dont un des auteurs du modèle `ResNet` a participer à

son développement [14].Le modèle `ResNext` de `detectron2` est entraîné sur les mêmes données que le modèle `ResNet` du même framework et obtient de meilleurs résultats pour la prédictions des objets [15]. Mais afin de comparer au mieux les jeux de données et les résultats, on gardera le même backbone pour chacun des jeux de données.

Transfert d'apprentissage : Pour ce qui est des jeux de données utilisées pour le transfert d'apprentissage, comme présenté dans la section 2, on utilisera les jeu de données COCO et Cityscapes [11, 12]. COCO contient de nombreuses images intérieures et extérieures dans un environnement d'Amérique du Nord. Les caractéristiques qui nous intéressent pour cette étude sont les caractéristiques que les modèles ont apprises par rapport à ces éléments extérieurs qui correspondent aux caractéristiques que l'on retrouve dans les images du projet. Le jeu de données Cityscapes quant à lui contient des composantes plus précises d'un environnement urbain comme les façades de bâtiments. Cependant, l'architecture des bâtiments peut être différents dus à la localisation des images du jeu. Les données de Cityscapes sont toutes prise dans des villes d'Allemagne.

Structure des données du projet : Dans ce projet, l'objectif initial du projet est de détecter et de placer 81 catégories différentes. De plus, le manque de données et le débalancement des classes sont des problématiques inévitables du projet. Afin de simplifier le problème, car les données n'étaient pas réalistes en comparaison de l'objectif demandé, le problème a été simplifié pour ne plus que d'avoir à détecter et à placer les 16 catégories les plus fréquentes d'un projet d'étude de fonds de prévoyance. Afin de réduire ce nombre de classe à 16, on a soit supprimés les classes les plus rares soit regroupé les catégories similaires entre elles.

En ce qui concerne les données de cette étude, on retrouve au total 9 238 objets annotés pour 2187 images. Presque la moitié de ces instances (47.5%) sont de la végétation.

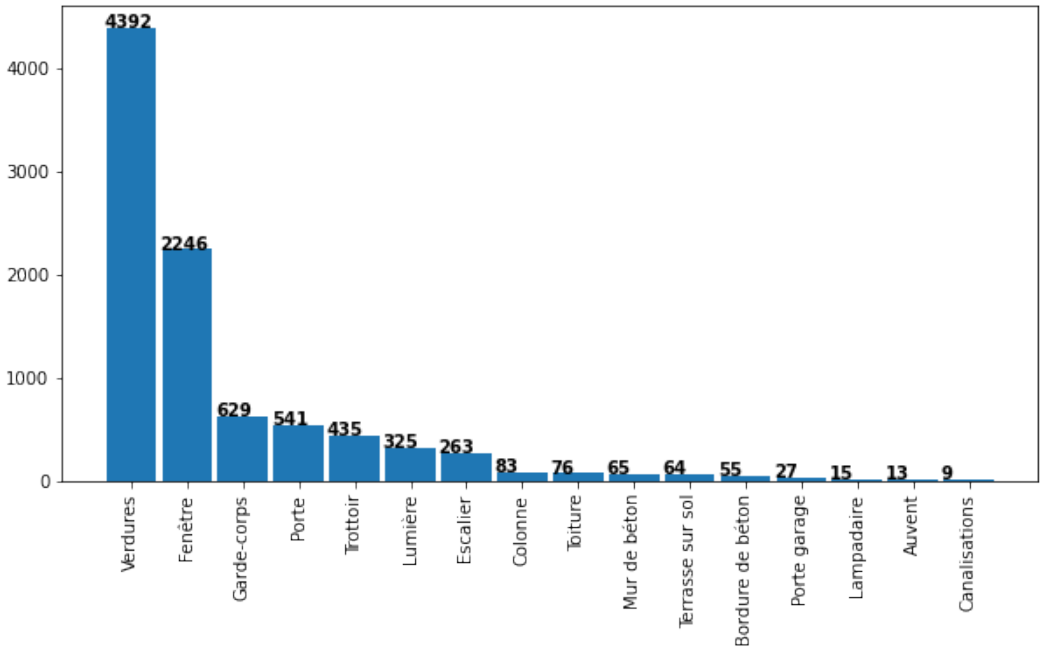

Figure 1: Distributions des données de l'étude en nombre d'instances d'objets par catégorie.

Métriques du projet : Les principales métriques utilisées pour comparer nos modèles seront la précision moyenne (AP) et l'intersection sur l'union (IoU), car dans le contexte du projet, on cherche à établir une liste des composantes présente sur la façade des bâtiments leurs positions exactes ainsi que leurs nombres ne nous intéresse donc pas.

Lors de la présentation des résultats, on verra apparaître les termes AP (la précision moyenne d'une classe), mAP (la précision moyenne de toutes les classes) et AP50 (la précision moyenne d'une

classe avec un IoU de 0.5 ou plus) l'IoU est l'intersection de l'aire prédite avec l'aire réelle diviser par l'union de ces deux aires.

3.1 Baselines

Les modèles de comparaison seront donc les suivants (les termes entre parenthèses sont les termes simplifiés désignant le modèle afin de raccourcir la présentation des résultats.) :

- `Mask R-CNN`: `ResNet50` + `FPN` avec transfert d'apprentissage sur le jeu de données `COCO` (`R50_COCO`)
- `Mask R-CNN`: `ResNet101` + `FPN` avec transfert d'apprentissage sur le jeu de données `COCO` (`R101_COCO`)
- `Mask R-CNN`: `ResNet50` + `FPN` avec transfert d'apprentissage sur le jeu de données `Cityscapes` (`R50_CITY`)

Dans les expérimentations de la section 4 nous comparerons donc les résultats des trois modèles décrit ci-dessus aux différentes méthodes étudiées:

- Rééchantillonnage des données
- Augmentation de données
- Focal loss

Baseline	mAP bbox	mAP50 bbox	mAP masque	mAP50 masque
R50_COCO	14.02	22.31	14.51	20.96
R101_COCO	14.35	22.29	14.35	19.38
R50_CITY	7.37	11.81	7.15	10.59

Table 1: **Segmentation d'instance :** mAP des classes de l'étude sur les différentes baselines de l'étude

En comparaison du modèle pré-entraîner sur Cityscapes les modèles pré-entraîner sur le jeu de donnée COCO obtiennent en moyenne une meilleure précision. Cependant, les backbones ne semble pour l'instant pas être déterminant sur la précision des modèles.

	Auvent	Lampadaire	Porte garage	Bordure de béton	Terrasse sur sol
R50_COCO	0.0	0.0	54.65	5.77	13.47
R101_COCO	0.0	0.0	36.01	10.98	19.36
R50_CITY	0.0	0.0	21.87	3.00	0.34

	Mur de béton	Toiture	Colonne	Escalier	Lumière
R50_COCO	0.00	6.03	3.37	14.86	16.81
R101_COCO	1.06	4.95	8.81	16.05	19.50
R50_CITY	0.00	0.59	2.11	9.74	11.05

	Trottoir	Porte	Garde-corps	Fenêtre	Verdures
R50_COCO	11.83	17.00	8.40	43.16	25.99
R101_COCO	13.83	20.59	6.60	44.25	27.53
R50_CITY	5.62	6.14	3.06	25.34	24.43

Table 2: **Segmentation d'instances :** AP pour chacune des classes de l'étude sur chaque baseline Les classes sont arrangées de la moins présente à la plus présente dans le jeu de l'étude. Figure 1 La classe "Canalisation" n'est pas présente dans le tableau, car dû à son nombre d'itération trop faible elle n'est pas dans le jeu de validation.

On remarque ici qu'une classe peu présente dans le jeu ressort tout de même avec de bons résultats cela peut s'expliquer dû à sa forme et sa taille. En général lors dans le cadre d'une étude de fonds de prévoyance, les clients prennent seulement cet élément précis en photos. Les porte de garages seront donc souvent en plein milieu de l'image et en grand.

4 Expérimentations

4.1 Synthèse des résultats

		mAP bbox	mAP50 bbox	mAP masque	mAP50 masque
R50_COCO	Échantillonnage	14.83	22.41	15.43	22.58
	Augmentation	7.40	13.84	8.49	12.98
	Focal loss	14.26	22.33	14.45	21.78
R101_COCO	Échantillonnage	15.21	22.44	15.31	21.78
	Augmentation	9.39	16.58	11.00	16.32
	Focal loss	15.00	22.81	15.25	22.37
R50_CITY	Échantillonnage	6.78	10.60	7.03	10.45
	Augmentation	4.66	9.43	6.12	9.37
	Focal loss	7.46	12.20	8.27	12.22

Table 3: Synthèse des résultats en mAP des classes de l'étude sur les différentes méthode de l'étude. Les meilleurs résultats de chaque backbone sont indiqué en gras.
Pour R50_COCO : Méthode de l'échantionnage avec un gain de 1.62% de précision moyenne avec un IoU $\in [0.5, 1]$ sur les masques.
Pour R101_COCO : Méthode du focal loss avec un gain de 2.99% de précision moyenne avec un IoU $\in [0.5, 1]$ sur les masques.
Pour R50_CITY : Méthode du focal loss avec un gain de 1.63% de précision moyenne avec un IoU $\in [0.5, 1]$ sur les masques.

		Auvent	Lampadaire	Porte garage	Bordure de béton	Terrasse sur sol
R50_COCO	Échantillonnage	0.0	0.0	54.98	13.60	15.15
	Augmentation	0.0	0.0	23.17	0.00	11.64
	Focal loss	0.0	0.0	55.74	6.82	14.51
R101_COCO	Échantillonnage	0.0	0.0	37.62	20.56	18.85
	Augmentation	0.0	0.0	23.17	0.00	11.64
	Focal loss	0.0	0.0	41.03	12.35	21.78
R50_CITY	Échantillonnage	0.0	0.0	27.75	1.80	1.12
	Augmentation	0.0	0.0	0.00	0.00	0.00
	Focal loss	0.0	0.0	21.88	5.92	2.89

		Mur de béton	Toiture	Colonne	Escalier	Lumière
R50_COCO	Échantillonnage	0.2	11.73	5.94	15.77	14.59
	Augmentation	0.0	0.0	0.28	2.49	14.43
	Focal loss	0.0	8.41	6.90	14.99	18.34
R101_COCO	Échantillonnage	2.07	3.59	8.15	18.28	18.69
	Augmentation	0.0	0.0	0.28	4.81	16.87
	Focal loss	0.0	6.54	9.41	16.63	15.47
R50_CITY	Échantillonnage	0.00	0.00	0.61	6.99	8.39
	Augmentation	0.0	0.0	0.00	1.59	9.79
	Focal loss	0.0	0.00	3.02	11.45	9.99

		Trottoir	Porte	Garde-corps	Fenêtre	Verdures
R50_COCO	Échantillonnage	11.52	18.17	4.91	43.33	27.24
	Augmentation	4.72	17.38	4.34	50.64	26.82
	Focal loss	10.49	17.16	5.66	43.89	25.29
R101_COCO	Échantillonnage	15.22	19.51	6.96	43.75	28.81
	Augmentation	5.67	19.99	8.11	51.58	26.66
	Focal loss	14.34	20.05	7.81	46.29	28.45
R50_CITY	Échantillonnage	5.30	5.66	2.92	23.70	23.12
	Augmentation	3.06	15.61	2.18	36.21	23.36
	Focal loss	5.30	7.49	3.27	24.58	23.62

Table 4: Résultats en AP (en %) de chacune des classes de l'étude pour chacune des méthodes d'**échantillonnage**, d'**augmentation des données** et de **focal loss**.
En vert, on retrouve les gains de précisions par rapport à la baseline.
En rouge, les pertes de précisions par rapport à la baseline.

4.2 Discussions

De manière générale, les méthodes pour contrer le débalancement de données apportent un gain en précision aux modèles. Cependant, ce gain reste faible et cela peut s'expliquer par le nombre d'instances présentes dans les classes. De plus, dépendement de la taille de l'objet, et de sa position dans les images d'entraînement et de validation, la précision peut fortement varier. C'est le cas pour les portes de garage, la précision est haute, mais cela est surtout dû au fait qu'en général les portes de garage sont au centre et en grand dans les images d'entraînement et de validation. Ceci peut se voir dans le tableau 4 dans les expériences d'augmentation de données. La précision de la classe baisse fortement, car l'augmentation de données va changer l'emplacement dans l'image de nos classes.

Échantillonnage : Dans cette étude, l'échantillonnage consiste simplement à rajouter naïvement un échantillon de l'annotation pour toutes les catégories qui représentent moins de 10 % du jeu de donnée. Cela équivaut à ajouter les instances de toutes les catégories sauf les deux les plus présentes (Les verdures et le fenêtres).

En comparant ces résultats, on remarque que les modèles `R50_COCO` et `R101_COCO` obtiennent en générale de meilleurs résultats que ceux de la Baseline. Notamment, des gains plus considérables pour les classes les moins présente. Au contraire, le modèle `R50_CITY`, lui perd en précision avec cette méthode. On remarque aussi que pour tous les modèles certaines classes sont toujours à 0% de précisions, ce sont des classes très peu présentes dans le jeu ou très complexe en forme et en positionnement dans l'image.

Augmentation de données : Dans cette expérience, lors de l'entraînement, les images utilisées pour l'entraînement subissent des transformations aléatoires : rotation ($\pm 90°$), modification de contraste afin de modéliser les aléas générés par les photos que les clients peuvent soumettre dans le contexte du projet.

Cette méthode demande plus d'entraînement pour converger, ceci s'explique par la quantité de « nouvelle données » ajouté par cette méthode. Par manque de ressources, nous n'avons donc pas pu obtenir de meilleurs résultats. De plus, l'augmentation a pour effet de modifier la position et la taille de nos instances de classes avec le redimensionnement et la rotation des images. Donc les classes spéciales qui nécessitent de prendre des photos spécifiques du bâtiment dans le contexte d'une étude de fonds perdent en précision, car instinctivement, si on nous demande de prendre une photo d'un objet, on va le centrer et le prendre en photo à l'endroit.

Focal loss : Cette expérience consiste à modifier la fonction qu'utilisent les modèles de baselines afin de pénaliser plus fortement les mauvaises prédictions lorsque le nombre d'annotations d'une classe est plus rare.

$$FL(p_t) = -\alpha(1-p_t)^{\gamma} log(p_t), \quad \alpha = 0.25, \gamma = 2$$

Cette méthode similaire à celle de l'échantillonnage permet un gain en précision pour tous les modèles sur les classes les moins représentées.

5 Conclusion

D'après les résultats de la section 4, on peut conclure que dans ce projet, utiliser un modèle pré-entraîner sur Cityscapes n'est pas intéressant vis-à-vis des modèles pré-entraînés sur COCO cela est dû a cause du style architecturale, car malgré le pré-entraînement sur un jeu plus orienté dans l'urbanisation, les modèles basés sur Cityscapes perdent beaucoup de précision par rapport à ceux pré-entraîner sur COCO. De plus, les différentes méthodes montrent des faibles gains en précision sur les différents modèles notamment sur les classes les moins représentées. Mais certaines classes ont toujours une précision trop faible voir nulle, cela est dû à leurs complexités ou à leurs rares présences dans le jeu de données. Afin d'obtenir de meilleurs résultats sur ces classes, il faut construire un jeu de données plus conséquent surtout si l'objectif à long terme est de détecter 81 classes. Avec les données actuelles même des méthodes plus personnalisées au jeu comme du post-traitement sur les détections ne serait pas suffisant pour répondre aux exigences des 81 catégories.

References

[1] Puja Bharati and Ankita Pramanik. Deep learning techniques—r-cnn to mask r-cnn: A survey. 2020.

[2] Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 2019.

[3] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *CoRR*, abs/1909.00169, 2019.

[4] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[5] Jitesh Jain, Jiachen Li, MangTik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. OneFormer: One Transformer to Rule Universal Image Segmentation. 2023.

[6] Chang Liu, Samad Sepasgozar, Sara Shirowzhan, and Gelareh Mohammadi. Applications of object detection in modular construction based on a comparative evaluation of deep learning algorithms. *Construction Innovation*, ahead-of-print, 05 2021.

[7] Isaac Osei Agyemang, Xiaoling Zhang, Isaac Adjei Mensah, Bernard Cobbinah Mawuli, Bless Lord Y. Agbley, and Joseph Roger Arhin. Enhanced deep convolutional neural network for building component detection towards structural health monitoring. In *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 202–206, 2021.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md`, 2019.

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[14] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016.

[15] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://paperswithcode.com/lib/detectron2/mask-r-cnn`, 2019.