AIN 311: Foundations of Machine Learning
2024 Fall Final Project Report
SportClassifier: Sports Image Classifier

Adam Sattout [1] **Mukhamediyar Amanzhol** [1]

Abstract

Sports image classification is a critical task in computer vision, with applications ranging from automated content tagging to action recognition in sports analytics. This study explores an improved pipeline for sports image classification by leveraging data augmentation and transfer learning techniques. Specifically, we employ horizontal inversion as a key data augmentation strategy to enhance the variability of training samples. In this study we use the popular 100 sports image dataset and our experiments demonstrate that transfer learning using a pre-trained ResNet50 model achieves the highest accuracy of 92.6%, significantly outperforming baseline methods.

1. Introduction

Sports image classification has emerged as a pivotal task in computer vision, enabling numerous applications such as sports analytics, automated video indexing, and the development of autonomous systems. The ability to accurately classify sports images facilitates advanced content understanding, enabling systems to recognize sports categories, players' actions, and relevant equipment. However, achieving high accuracy in this domain poses significant challenges due to the inherent diversity and complexity of sports datasets. Variability in scenes, dynamic player poses, lighting conditions, and overlapping features among sports classes further complicate this task, necessitating robust and scalable solutions.

Current efforts in sports image classification face limitations in identifying optimal models and hyperparameters capable of generalizing across diverse datasets. There is a critical need to systematically evaluate deep learning architectures and training strategies to achieve state-of-the-art performance while addressing issues such as overfitting, data imbalance, and computational efficiency.

To address these challenges, we propose a methodology centered on the following strategies:

- Incorporating data augmentation through horizontal inversion to enhance training data diversity and reduce class imbalance.
- Systematically testing and optimizing three different model architectures and evaluating their performance on sports image datasets.
- Leveraging transfer learning with pre-trained models such as ResNet50 and fine tuning the final classifier, to harness prior knowledge while adapting to the sports domain.

In this study, our objective was to compare the performance of multiple deep learning models in sports image classification tasks, to assess the impact of architectural choices and hyperparameter tuning on classification accuracy, to identify the most effective combination of augmentation strategies and model configurations to achieve state-of-the-art results. We will be using the most infamous dataset (100 Sports Image Classification) throughout our study.

2. Related Works

Image classification is and has been a fundamental task in computer vision for a long time, aiming to assign predefined labels to input images based on their visual content. It all started with simple edge detection and feature extraction from images. Then with advances in deep learning, many researchers started approaching the problem with different implementations of neural networks. An iconic study on the recognition of handwritten letters was done using basic CNN (Convolutional Neural Networks) with LeNet-5. In addition to that, models that came with the ImageNET Classification challenge conducted by Prof. Fei-Fei Li acheived great milestones in the field. For instance, in the model AlexNet was implemented, introducing the use of the ReLU activation function and dropout normalization technique. There is also VGG16 which introduced a new model that is 16 layers deep and has a top-5 accuracy of 92.7%. Furthermore, ResNet50 introduced the new technique of Residual Blocks which allowed for the model to be much deeper, thus allowing it to capture more details.

Although image classification methods and techniques have been utilized in sports for some time, recent breakthroughs in deep learning and computer vision have greatly improved their efficiency and range of applications. These technologies are now being applied across multiple facets of sports, such as analyzing player performance, preventing injuries, data labeling and enhancing fan engagement.

One notable example is the work of Zhang et al. (2020) which explored the use of image classification for player performance analysis in basketball. By employing deep learning techniques, they were able to identify player movements and categorize them into specific actions, such as shooting or passing. This research not only improved understanding of player dynamics but also provided insights for coaches to optimize training strategies.Another significant contribution is from Karpathy et al. (2014), which utilized convolutional neural networks (CNNs) to classify sports videos into different action categories. Their approach demonstrated that CNNs could effectively learn spatial and temporal features from video data, significantly improving classification accuracy compared to traditional methods.

3. Approach

3.1. The Dataset

The dataset used in this study is the Popular 100 Sports Image Dataset, a well-known benchmark for sports image classification tasks. This dataset contains images from 100 distinct sports categories in 224x224x3 resolution, capturing a wide variety of sports, ranging from common ones such as soccer and basketball to niche activities like fencing and pole vaulting. Each category comprises a diverse set of images, offering a comprehensive and challenging testbed for image classification models. The dataset is balanced, with an equal number of images per class, which ensures unbiased performance evaluation across all sports categories. The dataset contains approximately 15000 images with an average of 150 image per class, which can be a small number when comparing between 100 classes

To enhance the diversity of the training data and improve the generalization capability of the models, we applied horizontal inversion as a data augmentation technique. Horizontal inversion involves flipping an image along its vertical axis, producing a mirrored version of the original image, introducing variations in the spatial orientation of the subjects within the images.

For each original image in the dataset, we generated a horizontally inverted counterpart, effectively doubling the size of the training set. This augmentation technique preserves the semantic integrity of the images, as the flipped versions remain valid representations of their respective sports categories. For example, a left-handed tennis player in the original image appears as a right-handed player in the flipped image, but the action remains recognizable as tennis.

Figure 1. Examples from our dataset

By increasing the dataset's size and variability, horizontal inversion helps mitigate overfitting and ensures that the trained models can handle different spatial representations of sports actions. This approach also improves the models' ability to classify images with diverse orientations, thereby enhancing their robustness in real-world scenarios.

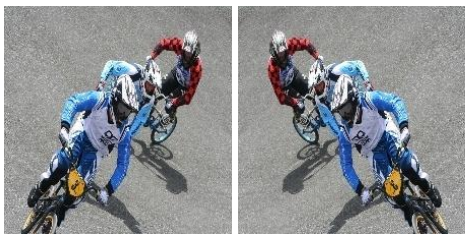

Figure 2. Inverted images examples

3.2. Techniques Used

3.2.1. Batch Normalization

Batch normalization (BN) is a technique introduced to improve the training of deep neural networks by addressing the issue of internal covariate shift. Internal covariate shift refers to the change in the distribution of inputs to a layer as the parameters of previous layers are updated during training. This can slow down the training process and make convergence more difficult. BN normalizes the input of each layer within a mini-batch by transforming it to have a mean of zero and a variance of one. It then applies learnable scale and shift parameters to allow the network to restore the original representational capacity if needed. Batch normalization provides several benefits:

- Stabilized Training: By normalizing activations, the optimization process becomes less sensitive to the choice of hyperparameters, such as the learning rate.
- Faster Convergence: Networks with BN converge faster since it mitigates the vanishing/exploding gradient problem.
- Regularization Effect: BN introduces some regularization, reducing the risk of overfitting.

3.2.2. Residual Blocks

Residual blocks allow for the construction of deeper networks by incorporating skip connections, which enable the gradient to flow more easily during backpropagation. These skip connections bypass one or more layers, allowing the input to be added directly to the output of a block of layers. This architecture helps in learning identity mappings, making it easier for the network to learn residual functions.

Residual blocks are a crucial architectural innovation in deep learning that effectively address the vanishing gradient problem, similar to techniques like batch normalization. The flexibility provided by residual blocks enables researchers and practitioners to experiment with deeper models, leading to enhanced feature extraction and representation capabilities. As a result, these models can capture more complex patterns in data, improving overall performance on various tasks while maintaining efficiency in training.

3.2.3. Transfer Learning

Transfer learning typically employs pre-trained models, which have already been trained on large datasets, allowing them to learn general features that can be useful for a variety of tasks. For instance, a model trained on a vast image dataset can be fine-tuned to recognize specific objects or categories in a smaller dataset with fewer examples.

The pre-trained model is used to extract features from the new dataset. After feature extraction, the pre-trained model can be fine-tuned on the new dataset. This usually involves retraining last few layers of the model by changing the weights, such that, process of feature extraction in initial layers does not change.

3.3. Model Architectures And Training Algorithm

To evaluate the effectiveness of deep learning for sports image classification, we explored four distinct neural network architectures. Each model represents a different design philosophy, ranging from simplicity and interpretability to deep, highly parameterized architectures capable of capturing complex patterns in the data. Below, we provide an in-depth analysis of each architecture and its role in this study.

3.3.1. SubCNN

This design of a Convolutional Neural Network (CNN) serves as a baseline model. It consists of two convolutional layers followed by two fully connected layers. The convolutional layers utilize ReLU activations and apply max-pooling to reduce spatial dimensions and computational cost. The fully connected layers aggregate the learned features into the final classification output.

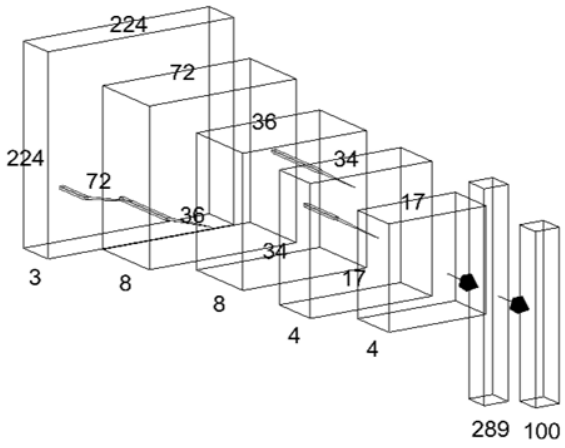

Figure 3. Architecture of our SubCNN model

This architecture was designed to balance simplicity and performance, providing a minimal yet functional structure to benchmark against larger, more complex models. While this model has limited capacity compared to deeper architectures, it serves as a proof of concept and allowed us to assess the effectiveness of different data augmentation strategies and to choose the best one between them.

3.3.2. AlexNetLike

AlexNet was one of the first CNNs to achieve groundbreaking performance in image classification tasks. In our study, we introduced a scalability factor to the original AlexNet design, a factor multiplied by the number of output channels in the convolutional layers to adjust them. The model is structured starting with five convolutional layers intercepted by max pooling layers. They progressively extract hierarchical features using filters with varying sizes, with earlier layers capturing low-level patterns like edges and gradients, and deeper layers focusing on more complex patterns specific to sports images. These are connected to 3 fully connected layers that aggregate features extracted by the convolutional layers and perform the final classification.

The scalability factor allows us to control the model's complexity, making this architecture versatile. By adjusting the

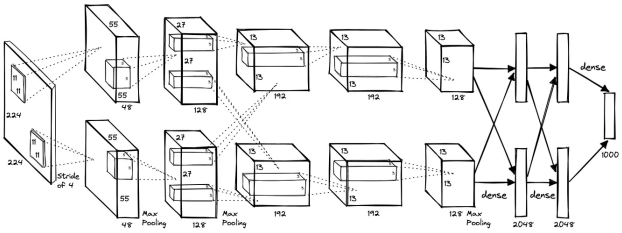

Figure 4. Architecture of AlexNet

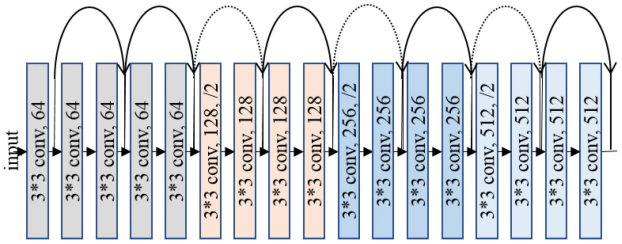

Figure 6. Architecture of ResNet

number of channels, we were able to explore the trade-offs between computational cost and model performance, providing insight into how much improvement can be gained by expanding the model without changing the structure.

3.3.3. VGG16LIKE

VGG16 is a deep architecture designed to emphasize simplicity and uniformity. It consists of 13 convolutional layers with Max-pooling layers in-between to downsample feature maps and 3 fully connected layers, using fixed 3x3 kernels with a stride of 1, which is smaller compared to earlier architectures like AlexNet. This small kernel size enhances the model's ability to learn fine-grained spatial patterns while maintaining a consistent receptive field, and the final 3 layers are densely connected and aggregate the learned features into the final class probabilities.

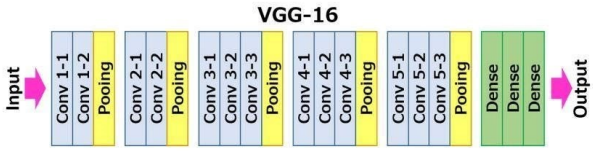

Figure 5. Architecture of VGG16 model

3.3.4. RESNETLIKE

ResNet50 is a deep residual network with 50 layers, designed to address the challenges of training very deep networks. Its key innovation is the introduction of residual blocks.

For this study, we employed transfer learning, leveraging ResNet50's pre-trained weights from ImageNet. This allowed us to take advantage of the network's ability to extract general visual features while fine-tuning the final classifier for the specific task of sports image classification. ResNet50's depth and architectural design make it particularly effective at capturing complex, hierarchical patterns in general, which also makes it a good idea to redesign it for sports.

3.4. Training Algorithm

The following algorithm outlines the steps for training a model using mini-batch gradient descent with Binary Cross-Entropy (BCE) loss. The algorithm involves creating batches, iterating through epochs and batches, performing forward and backward passes, and accumulating gradients.

Algorithm 1 Model Training Algorithm

Require: Dataset $\mathcal{D}$, model f, loss function $\mathcal{L}$, optimizer opt, number of epochs E, batch size B
Ensure: Trained model f_θ
Shuffle dataset $\mathcal{D}$ randomly
Split $\mathcal{D}$ into minibatches $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N\}$ of size B
for epoch $= 1$ to E **do**
  for each minibatch $\mathcal{M}_i$ in $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N\}$ **do**
    Extract inputs $\mathbf{X}_i$ and labels $\mathbf{y}_i$ from $\mathcal{M}_i$
    Reset accumulated gradients $\nabla_\theta \mathcal{L} \leftarrow 0$
    Forward Pass: Compute the model output $\hat{y} \leftarrow f_\theta(x)$
    Compute Loss: Calculate the Binary Cross-Entropy (BCE) loss $\mathcal{L} \leftarrow \text{BCE}(y, \hat{y})$
    Compute Gradients: Calculate gradients $\nabla_\theta \mathcal{L}$ with respect to θ
    Update model parameters: optimizer.step()
  end for
end for
Return Trained model f_θ

3.5. Model Optimization

To ensure optimal performance for the small CNN and AlexNet models, we conducted a systematic hyperparameter search. To make the tuning process computationally efficient, we used 30% of the training data. We performed a grid search over the following hyperparameters:

- Batch sizes: [16, 32]
- Learning rates: [0.1, 0.01, 0.001, 0.005]

- Expansion factors: [2, 4, 8, 12] (for controlling the number of output channels in convolutional layers).

The best combination of hyperparameters was selected based on the highest validation accuracy achieved during the search.

For the larger model VGG16, we adopted a more targeted approach to reduce computational overhead. Based on the stability and performance of AlexNet during tuning, we used the best learning rate identified for AlexNet as a starting point. We further refined the learning rate by reducing it during training in later epochs. Additionally, we also test the Adam optimizer with a learning rate of 0.0001. To accommodate the increased memory requirements of these models, we tested bigger batch sizes [32, 64].

For ResNet50, since tuning the last layer is a simple task we only tested a few settings that are reasonable while mentoring the stability of leraning

4. Results

4.1. Experiment Environment

The dataset was divided into training, validation, and test sets. For each sports category, 5 images were reserved for validation and another 5 for testing. The remaining images were used for training the models. All experiments were conducted using Google Colab, equipped with an NVIDIA T4 GPU. The software environment included Python 3.10.12 and PyTorch 2.5.1 with CUDA 12.1 support. Other essential python libraries such as NumPy, Matplotlib, pandas, and Scikit-learn were utilized for data preprocessing, visualization, and evaluation.

To assess the performance of the models, multiple evaluation metrics were employed:

- Accuracy: The proportion of correctly classified images out of the total test images. This metric provides a straightforward assessment of overall model performance.
- Precision: The proportion of true positives among all predicted positives for each class.
- Recall: The proportion of true positives identified out of all actual positives for each class.
- F1-Score: The harmonic mean of precision and recall, offering a balanced measure even when there are discrepancies between them.
- Training and Validation Loss Curves: These curves were analyzed to monitor the training process and identify signs of overfitting or underfitting.

4.2. Hyperparameter Tuning Results

When we tune our models, we observe in train loss versus time plots that higher learning rates and lower batch sizes had very hard oscillations when training, which is expected due to the learning stage not being stable in these conditions. We also opt to using ReLU activation function since it performs well avoiding problems of vanishing/exploding gradients.

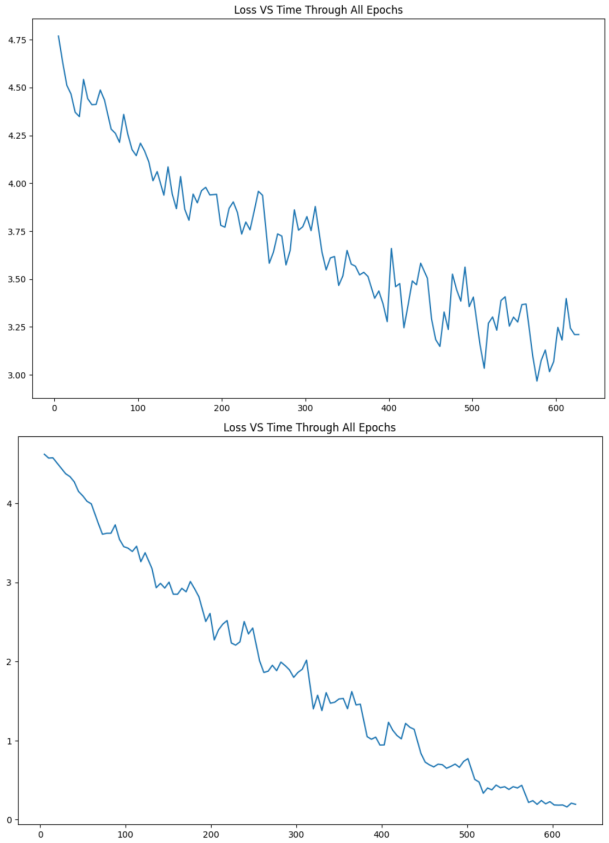

Figure 7. Low Batch Size AND High Learning Rate Versus High Batch Size AND Optimal Learning Rate, Respectively

Increasing expansion factor beyond 8 made SubCNN worse, while 12 was the best performance for AlexNetLike. This may indicate that a higher expansion may have better results. However, we suspect that it will make a huge increase due to increments in performance being little compared to the size difference.

For VGG16 Adam optimizer performed with a 5% increase in accuracy with a more stable learning process in general. We can see all other hyperparameters settings we used in Table 1.

Table 1. Hyperparameter settings for all models

Model	Optimizer	Learning Rate	Batch size	Expansion Factor	Epochs	Parameters Count
SubCNN	SGD	0.005	32	8	10	285K
AlexNetLike	SGD	0.005	32	12	15	10.6M
VGG16Like	Adam	0.0001	64	-	15	138M
TunedResNet	Adam	0.001	32	-	10	25.6M

Table 2. Classification accuracies for naive Bayes and flexible Bayes on various data sets.

MODEL	ACCURACY	F1
SUBCNN	31.6%	29.5%
ALEXNETLIKE	66%	64%
VGG16LIKE	79.3%	78%
TUNEDRESNET	92.6%	92.46%

4.3. Model results

We can see our results in Figure 15. As expected, the ResNetLike model did the best with 92.6% accuracy. If we look at training-validation loss curves for the models we built from the scratch up we can compere between them and see how making the model more complex and adding more layers can enhance the general leraning experience. The AlexNetLike model starts to overfit much earlier than VGG16 due to it not being able to assess as much features as VGG16 does (figure 8). However, this does not mean parameter size is everything, because ResNet has done a better job than VGG16 while having much less parameters.

Figure 8. Football

Figure 9. Hurdles

Figure 10. Judo

Figure 11. Sky Surfing

Figure 12. Horseshoe Pitching

Figure 13. Ultimate

Figure 14. Some samples missclassified by TunedResNet and their predicted labels

If we take a closer look at what areas our ResNetLike model failed, we will see some patterns. For instance, it struggles with images that contain multiple people like a whole team of cheerleaders or a full crowd. It can also miss between similar sports for example different types of snow activities or 1 to 1 combat sports. Also it cant distinguish sports that have a very similar concept and usually have very close setups like different sports of throwing objects like javalins or pitching.

5. Conclusion

This study explored the performance of four distinct deep learning architectures—Custom SubCNN, AlexNetLike, VGG16, and TunedResNet50—on the task of sports image classification. Through a comprehensive evaluation of these models, we identified ResNet50 with transfer learning as the best-performing architecture. Its ability to leverage pre-trained features combined with fine-tuning for our dataset resulted in superior accuracy

Some key takeaways:

- Increasing model's size without changing anything else is usually not the right answer, as seen with AlexNet-Like scaling factor and the jump from VGG16 to ResNet.
- Deeper architectures like VGG16 and ResNet50 outperformed shallower networks, emphasizing the importance of model depth and hierarchical feature extraction.
- Transfer learning proved to be a powerful approach for achieving high accuracy with minimal computational resources compared to training from scratch.

Deeper architectures like VGG16 and ResNet50 outperformed shallower networks, emphasizing the importance of model depth and hierarchical feature extraction. Transfer learning proved to be a powerful approach for achieving high accuracy with minimal computational resources compared to training from scratch. The scalability factor in AlexNet provided valuable insights into the trade-offs between model complexity and performance.

To build on the findings of this study, we propose the following directions for future research:

- A hierarchical classification approach could group sports into broader categories (e.g., team sports, ball sports, water sports) before further classifying them into specific sports (e.g., soccer, basketball, swimming). This hierarchical structure may improve classification accuracy by enabling models to focus on distinguishing finer-grained details within smaller subsets of related classes.
- More advanced techniques like SE Attention or spatial attention could be used. As well as newer model architectures.
- The direction of the study could be changed to more specific events in sports like deciding whether a pass in football was offsite or a punch in boxing was illegal.

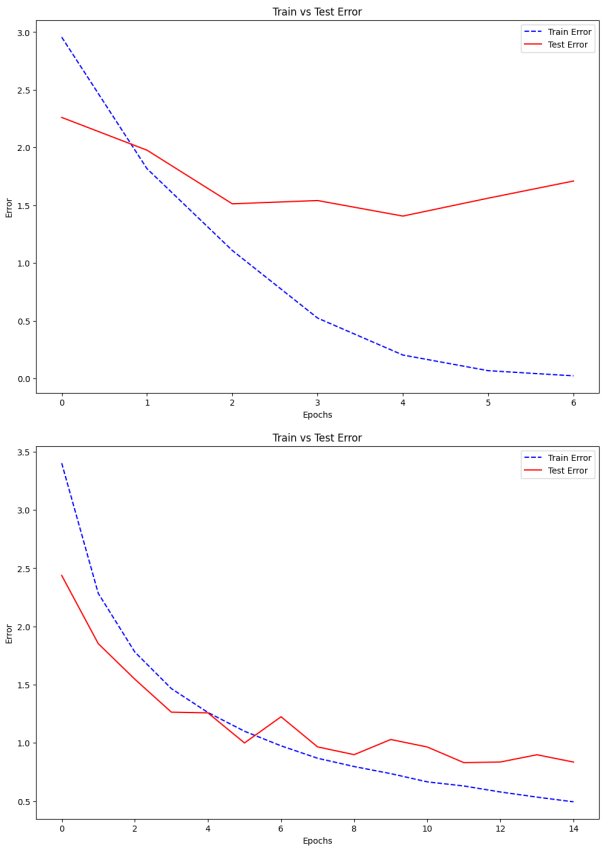

Figure 15. AlexNetLike VS VGG16Like, Respectively

References

Author, A. 100 sports image classification. *Kaggle*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Zhang, Y., Liu, Y., and Wang, X. Action recognition in basketball using deep learning techniques. *International Journal of Sports Science Coaching*, 2020.