CLIP-based Zero-shot classificator
ATDL project report

Matteo Monti
Politecnico di Milano
matteo6.monti@mail.polimi.it

Samuele Mariani
Politecnico di Milano
samuele1.mariani@mail.polimi.it

Abstract

In an era characterized by the rapid fusion of visual and textual data, the development of models capable of comprehending and interacting with both modalities has become paramount. One such pioneering model is CLIP (Contrastive Language-Image Pre-training), which excels in capturing non-trivial relationships between images and natural language. In this report we outline the development and training of a dual-encoder model based on CLIP, further showcasing its capabilities through the construction of a zero-shot classifier. We also provide a comparative view of the performance of our locally-trained model, used for zero-shot classification, against those of the pretrained CLIP model made available by OpenAI, thus exhibiting the potential of an optimally trained model.

1. Introduction

CLIP is a revolutionary vision-language model that has been pre-trained on a vast corpus of text and image data. By leveraging a shared representation space, CLIP can seamlessly connect images and text, enabling it to understand the semantic content of both modalities. Its architecture, rooted in contrastive learning, equips it with the ability to perform tasks like image classification, textual image retrieval, and more without the need for task-specific fine-tuning. Our aim was to successfully implement and train a model based on the original CLIP architecture and showcase one of its possible applications.

In section 3 of this report show the process of implementing a dual encoder model inspired by the CLIP architecture. We provide a comprehensive overview of the model's constituent components and offer insights into the specifics of the training phase.

Sections 4 and 5 will be dedicated to showcasing two applications of our model to real-life problems. In section 4 we explore the development of an *Image Retrieval* model, with which a user prompts a query and gets back some images that corresponds to the query, while section 5 focuses on the *Zero-shot Classification* problem, that is, perform classification on unseen images by using the CLIP model without the need to re-train or fine tune it.

Lastly, Section 6 presents the outcomes of our Zero-shot Classifier, including a comparative analysis of its performance against a powerful pre-trained model.

2. Related work

Our work draws significant inspiration from the groundbreaking research presented in the original CLIP (Contrastive Language-Image Pre-training) paper, authored by Radford et al. [4]. CLIP introduced a transformative vision-language model that leverages a shared representation space to connect text and image representations. This pioneering model can understand the semantic content of both images and text, allowing it to perform a wide range of tasks, including image classification and textual image retrieval, without the need for task-specific fine-tuning.

3. Proposed approach

3.1. CLIP architecture

CLIP is built on a dual-encoder architecture, where it has separate encoders for text and vision. CLIP's main idea is to map text and image inputs into a shared embedding space such that semantically similar text and image pairs have embeddings that are close together in this shared space.

3.1.1 Dual Encoder structure

The dual encoder consists of two main components: a text encoder and a vision encoder. In particular we used two powerful pre-trained encoders, and we made their outputs have the same size of 256 units in order to compute the loss between the two representations.

Text Encoder The text encoder takes a text description as input and converts it into a fixed-length embedding vector. We chose to use BERT [2], a powerful transformer-based NLP model developed by Google. Since the text encoder needs to be able to capture the semantic content of the input sentence, the choice of a transformer-based model is suitable to the problem. In fact, thanks to their attention mechanism, transformers can learn on which section of the input to focus the most, resulting in extraordinarily good performances on NLP tasks.

Vision Encoder The vision encoder takes an image as input and converts it into a fixed-length embedding vector as well. We chose to use XCeption [1], a deep convolutional neural network (CNN) architecture, which has been developed to improve the Inception architecture both in efficiency and performance terms. The vision encoder has the role of capturing the visual information present in the image and encode it as a dense vector.

Projection Head After obtaining text and image embeddings, a projection head is applied to both embeddings to project them into a common shared embedding space. Our projection head consists of a few layers with non-linear activation functions. Its role is to align the text and image embeddings by projecting them in the shared space while preserving their semantic information. Keeping in mind our limited computational power, we chose a shared space composed of 256 features, as we have noticed that this value provides a good trade-off between information loss and computational cost.

3.1.2 Contrastive Learning and Contrastive Loss

The key idea behind CLIP is contrastive learning. In contrastive learning, the model is trained to bring positive pairs (text-image pairs that should be semantically similar) closer in the shared embedding space while pushing negative pairs (text-image pairs that should be dissimilar) farther apart. During training, CLIP is presented with pairs of text and images. These pairs can be labeled as positive (semantically matching) or negative (semantically mismatching). A loss function, called *contrastive los*s, is used to measure the similarity between the embeddings of the text and image in the shared space. The loss encourages positive pairs to have high similarity scores (i.e., be close in the embedding space) and negative pairs to have low similarity scores (i.e., be distant in the embedding space). By training on large-scale datasets containing text-image pairs with contrastive loss, CLIP learns to understand the relationship between text and images in a way that allows it to perform various tasks, such as *text-based image retrieval* and *image classification*, without task-specific fine-tuning.

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Figure 1: The contrastive loss as proposed in the original paper

3.2. Model Training

In this section we will detail the training phase as well as the choices that characterized it.

The Dataset In order to train our model, we chose to employ the well known MS-COCO (Common Objects in Context) [3] dataset. Our choice was guided by the need to train the model on a sizable corpus of images and their relative descriptions; MS-COCO is a large-scale image recognition dataset frequently used for object detection, segmentation, and captioning tasks. It contains over 80,000 images annotated with 5 different captions each. To train the model we have repurposed the image-captions pairs to respectively train the visual encoder and the text encoder.

TFRecords To ensure a more efficient retrieval and storing of data we have resorted to TensorFlow's TFRecord format, which stores data in a serialized and binary fashion, that can be easily read and written using TensorFlow's tf.data API.

Local Training The local training was conducted using a single NVIDIA rtx 3060 GPU. We trained the dual encoder on a 30,000 images training set (37,5% of the dataset) using 3 captions per image, thus producing 90,000 image-caption pairs. Using a validation set of 6,000 images, we trained for 8 epochs with a batch size of 256. Each epoch took an average of 4,250 seconds (70.8 minutes) to complete, amounting to a cumulative training

Table 1: Model's hyperparameters

Hyperparameters	Value
Projection Dimensions	256
Batch size	256
Epochs	8
Captions per Image	3
Validation split	0.2
Dropout rate	0.1
Optimizer	AdamW
Learning rate	0.001
Weight decay	0.001
Early stopping patience	5
Reduce LR on Plateau patience	3
Reduce LR on Plateau factor	0.2

time of 9.4 hours. We used *AdamW* as optimizer with `learning_rate = 0.001`, `weight_decay=0.001` and setting a `dropout_rate=0.1` for both encoders this was the training loss we obtained. For the complete hyperparameters' list, please refer to Table 1

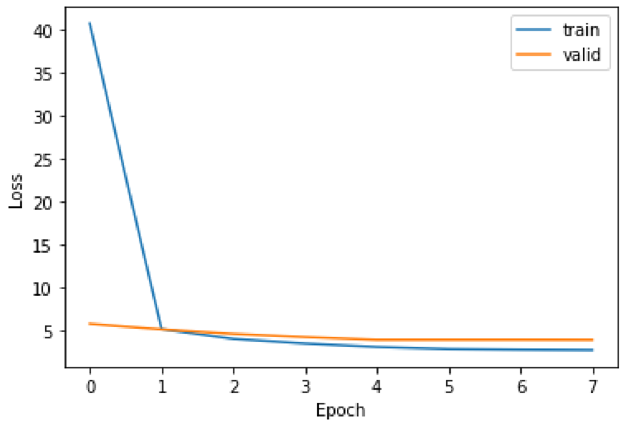

Figure 2: plot of both training loss (in blue) and validation loss (in orange) across the 8 training epochs

4. Image Retrieval

One of the most straightforward applications of a CLIP model is **image retrieval**, which is the process of searching and retrieving images from a large database basing on a user-made natural language query. In our case we chose to use textual queries, in this way we were able to test the effectiveness of both the visual and textual encoders.

Usage The user prompts a query, in the form of a short sentence, which describes the kind of images she wants to get, the system searches for the K most similar images in its database and retrieves them to the user.

Functioning Once the user has provided the query, the text encoder is used to get its embedding. Then, we compute its similarity with respect to the embeddings of each image in the database, and we retrieve the K most similar ones to the user. In this work we made use of the dot similarity, which is simply the dot product of the two embeddings' vectors, but there are other valid similarity measures such as the cosine similarity or the euclidean distance.

Results To showcase our model performances on the Image Retrieval task, we provide two queries and their respective results. In Figure 3 it's possible to see the resulting images for query *Cars on the street at night*, while in Figure 4 are provided the resulting images for query *A beautiful view of a natural landscape in the countryside*. We tried experimenting with various prompts and found out how specific and detailed prompts seem to generate better overall results, while short and generic ones often lead to uncorrelated resulting images.

5. Zero-Shot Classifier

After having successfully implemented and trained a CLIP-based dual encoder model, and having tested it in the straightforward task of image retrieval, we set out to showcase the versatility of this model by stepping up to the more complex task of Image classification, specifically Zero-Shot image classification.

5.1. Motivations

State of the art computer vision models are characterized by a limited understanding of the visual world based on their training data. Generally speaking, such models are able to achieve competitive performances on specific tasks (e.g. Image classification) and datasets, but they fail to generalize and achieve similar performances when presented with new classes or images that go beyond the scope of the dataset they have been trained with.

It is easy to see how this issue could translate in a real-world problematic: building classifier for particular and specific use-cases (e.g. defect detection in product-line quality control) could in fact become extremely hard when it is not possible to gather enough data to fine-tune a CV model with traditional methodologies or when dealing with rapidly changing environments where new classes may emerge frequently. Models such as CLIP allow to partly solve these issues by showing considerable flexibility when performing tasks across datasets different from the one they have been trained on, often requiring zero retraining. Zero-Shot Classification refers to the paradigm which involves a model able to recognize objects that it has never seen before. Specifically, Zero-Shot image classification involves

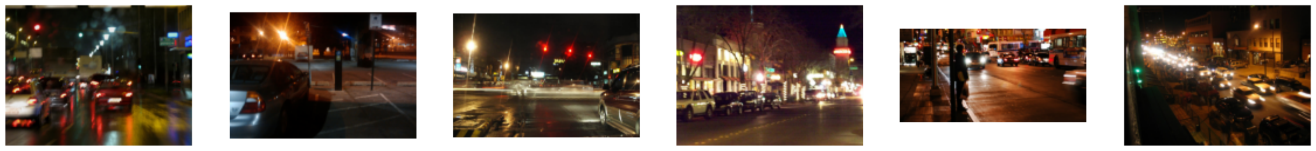

Figure 3: Resulting images for query *Cars on the street at night*

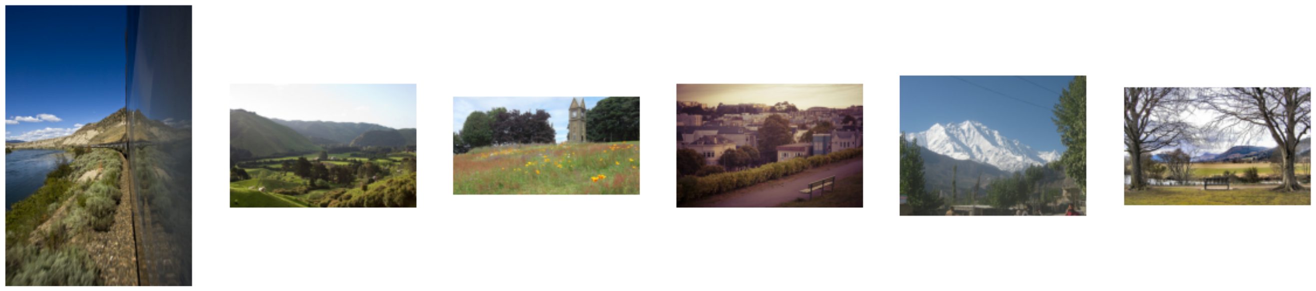

Figure 4: Resulting images for query *A beautiful view of a natural landscape in the countryside*

being able to assign images to different categories using a model that is not specifically trained on a dataset containing labeled examples belonging to those categories.

5.2. Implementation

In the following section we will detail the implementation of a Zero-Shot image classifier based on a Dual Encoder model. The visual and text encoders create a shared 256-dimensional vector space where features from both images and textual data are projected. To allow the model to assign a class to each image we first need to convert the usual image labels that we're familiar with into descriptive text sentences. Given the class 'dog', its corresponding descriptive text sentence will be "A photo of a dog". Each one of the so obtained new classes is given as input to the text encoder that outputs its corresponding encoded vector $T_1, T_2, T_3 \dots T_{10}$ respectively. As depicted in Figure 5, given an image that needs to be classified, we encode it with the visual encoder obtaining its corresponding vector I_1, then we compute the similarity between vectors $T_1, T_2 \dots T_{10}$ and I_1 by computing their cosine similarity. Finally we select the class with the highest score. That class will be the label predicted by the model for that specific image.

6. Experiments

We evaluated the performance of our Zero-shot Classifier on two distinct datasets and conducted a comparative analysis with a highly proficient pre-trained variant of CLIP. This approach allowed us to benchmark our model against one of the top-performing CLIP models currently available.

6.1. Datasets

Custom dataset our custom dataset comprises 9,300 images categorized into 10 classes: fruit, motorbike, person, flower, car, airplane, cat, food, house, and dog. Given that this dataset includes everyday objects, we anticipate that our classifier will exhibit strong performance, considering that the CLIP model has been trained on COCO, a dataset that also encompasses a wide range of common objects.

We opted for a custom dataset primarily due to the unavailability of a pre-existing dataset that precisely aligned with our requirements for common objects. Our approach involved initially selecting the specific classes of objects we desired and subsequently scouring Kaggle to obtain individual datasets for each of these selected classes. This method allowed us to curate a dataset that met our specific needs and objectives.

Animals-10 This dataset encompasses a total of 28,000 animal images distributed across 10 distinct categories: dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, and elephant. This dataset presents a more challenging task as it exclusively concentrates on animals, thereby assessing our classifier's capacity to generalize across potentially unseen or under-represented animal species. Consequently, we anticipate that the performance metrics on this dataset will likely be lower compared to the previous dataset, which included more common objects.

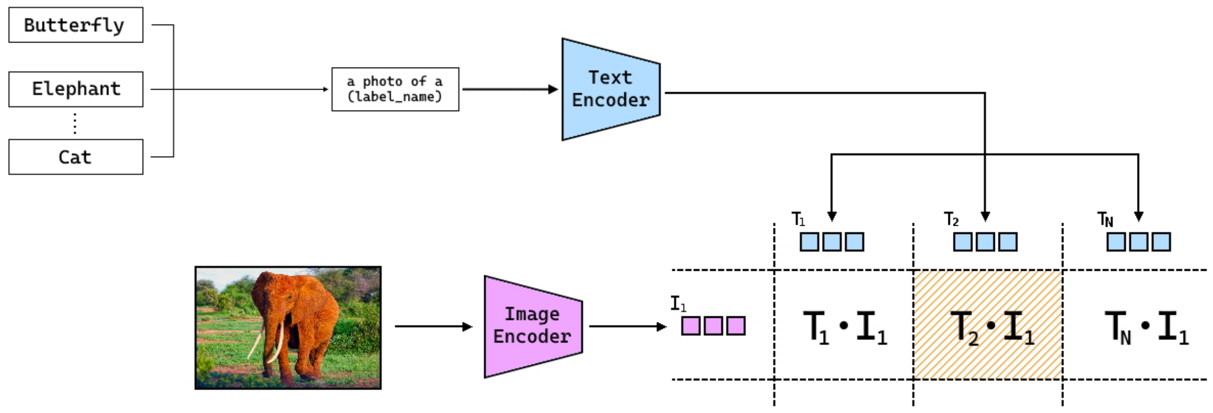

Figure 5: Depiction of Zero-Shot classification underling idea, both the image and the captions are projected onto the embedding space, then the cosine similarity is computed between the vectors of both the image and the textual class labels. The label encoding which provides the highest similarity will correspond to the predicted class.

6.2. Experiments setup.

In Zero-shot Classification, by its very nature, there is no need for training on the specific task. Hence, we allocated 100% of the available images to serve as the validation set. We conducted experiments using both the pre-trained CLIP models and our imported CLIP models on the datasets mentioned earlier, gathering comprehensive results for analysis.

6.3. Xception and EfficientNetB7

In order to compare the performance on the classification task conducted on the Animals-10 dataset of our model with those of conventional methods, we decided to locally train two additional models by fine-tuning Xception and EfficientNetB7 from the available models from `keras.applications`. We chose Xception since it is the same architecture that we used as vision encoder for our model; additionally, training EfficientNetB7 seemed to us a sensible choice given its competitive performances in classification tasks.

6.4. Results and discussion.

Regarding model performance As depicted in Table 2, our model achieves commendable performance on our custom dataset. However, there is a noticeable decline in performance when transitioning to the Animal-10 dataset. Nonetheless, the results on Animal-10 remain acceptable. This performance variation can largely be attributed to our training settings. We trained our CLIP model using a subset comprising 37.5% of the available images, and this training lasted for only 8 epochs. This choice was driven by the constraints of our computational resources, necessitating a balance between performance and overall computational cost. We notice how both Xception and EfficientNetB7 fail to achieve the same performances of CLIP despite having been specifically trained on the Animals-10 dataset.

Impact of the pre-trained model Notably, performance experiences a substantial boost when employing the pre-trained CLIP model provided by OpenAI. This model was trained on a larger dataset, encompassing higher-quality images and undergoing a longer training duration. It's worth noting, however, that even with the pre-trained model, performances on Animal-10 are marginally inferior to those on our custom dataset. This disparity can be attributed to the more specialized nature of Animal-10. For a comprehensive view of the results, please refer to Figures 6 and 7, which provide the complete confusion matrices generated during the experiments.

7. Conclusion

In our work, we explored the potentialities of CLIP's (Contrastive Language-Image Pre-training) paradigm and its flexibility. Through two practical applications, image retrieval and zero-shot classification, we demonstrated the versatility and adaptability of this architecture. Finally we compared the overall performances of our locally-trained model and those of a pre-trained CLIP against those of two commonly used architectures (Xception and EfficientNetB7), showing how the pretrained CLIP is able to outperforms them both.

Table 2: Experimental results

Target Dataset	Statistics	Xception	EfficientNetB7	Our Model	Pre-trained Model
Custom Dataset	Test Accuracy (%)	-	-	85.75	**99.09**
	Precision (%)	-	-	89.63	**99.15**
	Recall (%)	-	-	86.41	**99.07**
	F1 score (%)	-	-	86.66	**99.10**
Animal-10	Test Accuracy (%)	44.22	97.47	59.02	**98.73**
	Precision (%)	-	-	62.49	**98.41**
	Recall (%)	-	-	62.30	**98.72**
	F1 score (%)	-	-	55.86	**98.56**

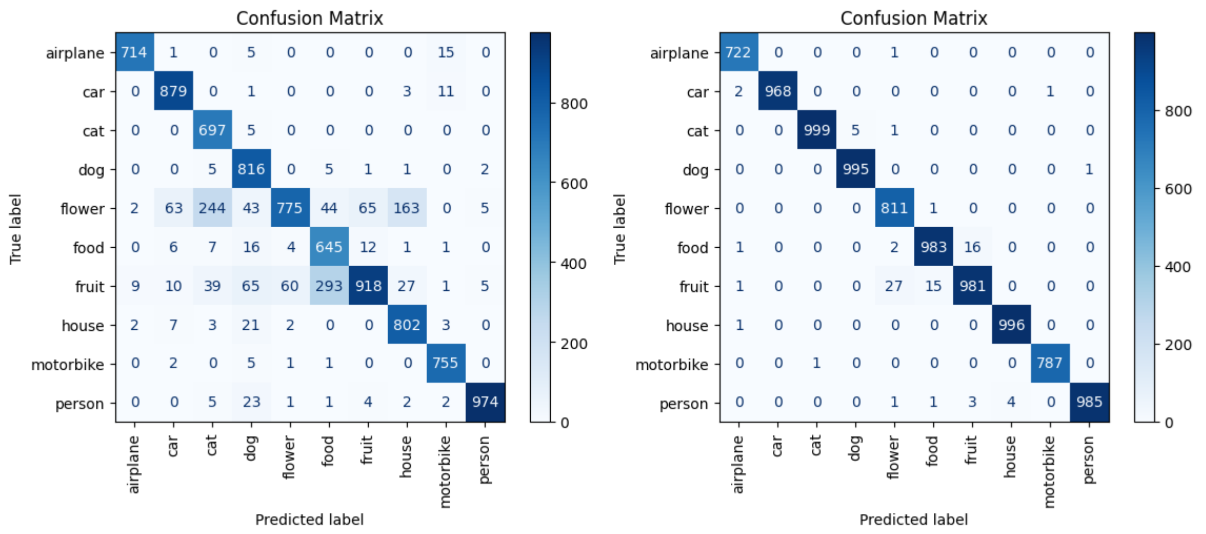

Figure 6: Performance comparison on the *Custom Dataset*, our model is on the left, the pre-trained model is on the right

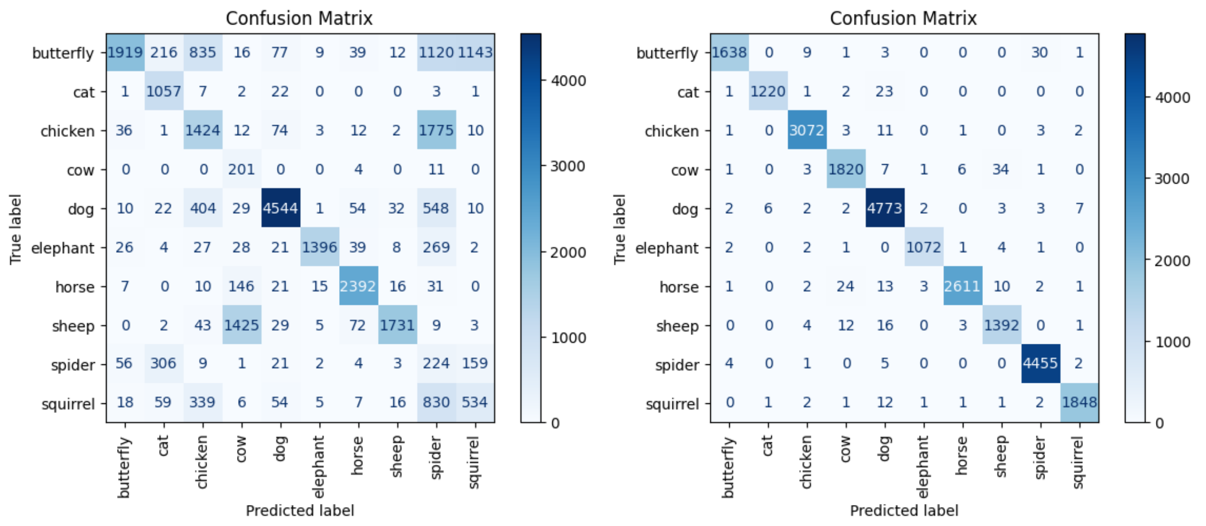

Figure 7: Performance comparison on *Animal-10*, our model is on the left, the pre-trained model is on the right

References

[1] F. Chollet. Xception: Deep learning with depthwise separable convolutions, 2017. 2

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 2

[3] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015. 2

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021. 1