Report

- Main objective of the analysis that also specifies whether your model will be focused on a specific type of Deep Learning or Reinforcement Learning algorithm and the benefits that your analysis brings to the business or stakeholders of this data.
- Brief description of the data set you chose, a summary of its attributes, and an outline of what you are trying to accomplish with this analysis.
- Brief summary of data exploration and actions taken for data cleaning or feature engineering.
- Summary of training at least three variations of the Deep Learning model you selected. For example, you can use different clustering techniques or different hyperparameters.
- A paragraph explaining which of your Deep Learning models you recommend as a final model that best fits your needs in terms of accuracy or explainability.
- Summary Key Findings and Insights, which walks your reader through the main findings of your modeling exercise.
- Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model or adding specific data features to achieve a better model.

You can run my notebook on Kaggle. GPU recommended:

https://www.kaggle.com/konutech/cifar100-image-classification-to-1-of-100-classes

1. Main objective of the analysis that also specifies whether your model will be focused on a specific type of Deep Learning or Reinforcement Learning algorithm and the benefits that your analysis brings to the business or stakeholders of this data.

The main objective of the analysis was to classify images with highest possible accuracy.
In my project the images can be classified by 100 classes, e.g. "mountain", "road". Due to characteristics of the challenge I have used few variations of Convolutional Neural Networks (or CNN).
Simplest benefit from implementation of image classification automation for a business area might be a case when a user is expected to upload a particular picture but instead he uses a random one. Let's imagine that insurence company expects to upload an image of a car but receives a picture of a horse.

2. Brief description of the data set you chose, a summary of its attributes, and an outline of what you are trying to accomplish with this analysis.

The data I used comes from https://www.cs.toronto.edu/~kriz/cifar.html
In contrary to data used during the course which is grouped by 10 classes, and for the sake of studying deep learning methods, I decided to increase the challenge of classification to 100 groups. The dimensions of training set are (50000, 32, 32, 3) so in other words:

- the model was trained using 50.000 classified images
- the images are in RBS scale and each one has 32*32 pixels

Test set consists of 10.000 images.

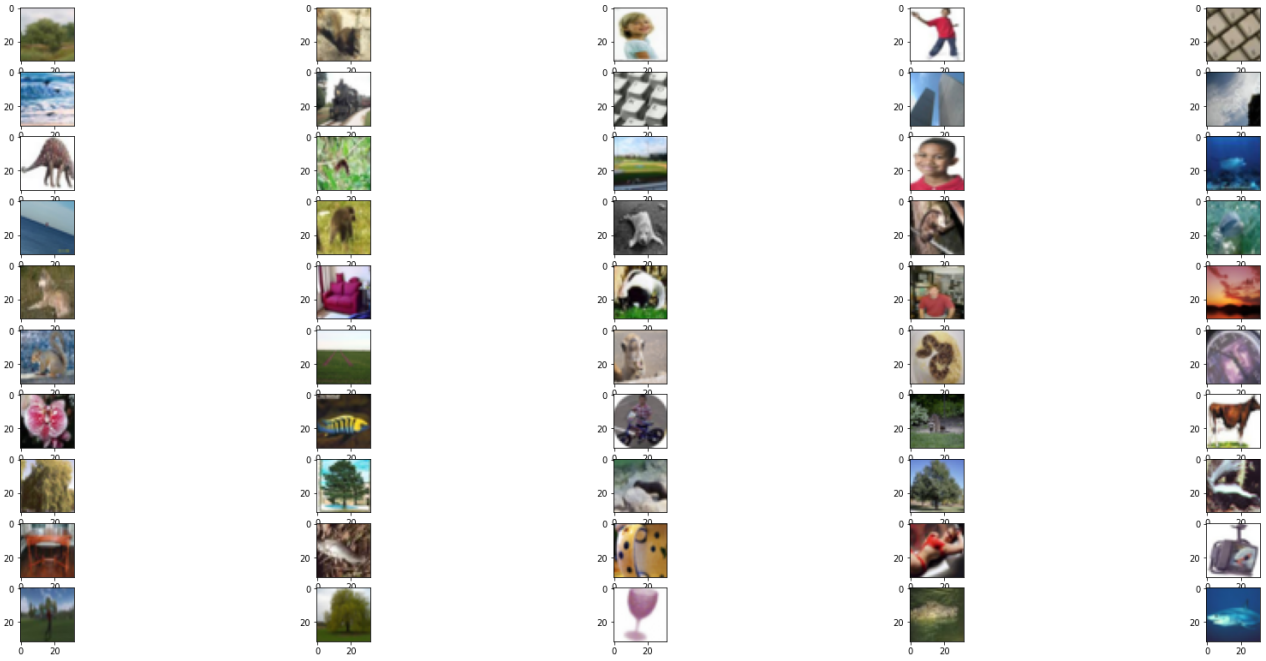
The labels are:

labels_dictionary =\ {0: 'apple', 1: 'aquarium_fish', 2: 'baby', 3: 'bear', 4: 'beaver', 5: 'bed', 6: 'bee', 7: 'beetle', 8: 'bicycle', 9: 'bottle', 10: 'bowl', 11: 'boy', 12: 'bridge', 13: 'bus', 14: 'butterfly', 15: 'camel', 16: 'can', 17: 'castle', 18: 'caterpillar', 19: 'cattle', 20: 'chair', 21: 'chimpanzee', 22: 'clock', 23: 'cloud', 24: 'cockroach', 25: 'couch', 26: 'crab', 27: 'crocodile', 28: 'cup', 29: 'dinosaur', 30: 'dolphin', 31: 'elephant', 32: 'flatfish', 33: 'forest', 34: 'fox', 35: 'girl', 36: 'hamster', 37: 'house', 38: 'kangaroo', 39: 'computer_keyboard', 40: 'lamp', 41: 'lawn_mower', 42: 'leopard', 43: 'lion', 44: 'lizard', 45: 'lobster', 46: 'man', 47: 'maple_tree', 48: 'motorcycle', 49: 'mountain', 50: 'mouse', 51: 'mushroom', 52: 'oak_tree', 53: 'orange', 54: 'orchid', 55: 'otter', 56: 'palm_tree', 57: 'pear', 58: 'pickup_truck', 59: 'pine_tree', 60: 'plain', 61: 'plate', 62: 'poppy', 63: 'porcupine', 64: 'possum', 65: 'rabbit', 66: 'raccoon', 67: 'ray', 68: 'road', 69: 'rocket', 70: 'rose', 71: 'sea', 72: 'seal', 73: 'shark', 74: 'shrew', 75: 'skunk', 76: 'skyscraper', 77: 'snail', 78: 'snake', 79: 'spider', 80: 'squirrel', 81: 'streetcar', 82: 'sunflower', 83: 'sweet_pepper', 84: 'table', 85: 'tank', 86: 'telephone', 87: 'television', 88: 'tiger', 89: 'tractor', 90: 'train', 91: 'trout', 92: 'tulip', 93: 'turtle', 94: 'wardrobe', 95: 'whale', 96: 'willow_tree', 97: 'wolf', 98: 'woman', 99: 'worm'}

Every 10.000th image of Traing set

In [87]:
```
images_to_show = []
for i, image in enumerate(X_train):
    if (i + 1) % 1000 == 0:
        images_to_show.append(i)
```

In [88]:
```
plt.figure(figsize=(30,15))
columns = 5
for i, image in enumerate(X_train[images_to_show]):
    plt.subplot(len(images_to_show) / columns + 1, columns, i + 1)
    plt.imshow(image)
```



3 Brief summary of data exploration and actions taken for data cleaning or feature engineering.

The data set provided did not needed any actions related to data cleaning.
In case of "feature engindeering" some action were taken known as Image Augmentation.
Basically Image Augmentation provides solutions to increase the size of training set (provides new sample of images) due to simple changes in atrtibutes of images.
The simplest examples of Image Augmentation are:

- vertical copies of images
- shifts in orientation of images
- rotations of images by some degrees

4 Summary of training at least three variations of the Deep Learning model you selected. For example, you can use different clustering techniques or different hyperparameters.

- Model 1 scored:
- Model 2 scored: 0.6207
- Model 3 scored:

Accuracy report for Model 1: 0.4711

```
              precision    recall  f1-score   support
```

0	0.73	0.70	0.71	100
1	0.56	0.65	0.60	100
2	0.34	0.23	0.28	100
3	0.20	0.17	0.18	100
4	0.31	0.20	0.24	100
5	0.37	0.46	0.41	100
6	0.46	0.50	0.48	100
7	0.64	0.48	0.55	100
8	0.71	0.55	0.62	100
9	0.56	0.68	0.61	100
10	0.49	0.33	0.40	100
11	0.33	0.30	0.32	100
12	0.50	0.52	0.51	100
13	0.39	0.48	0.43	100
14	0.44	0.43	0.44	100
15	0.42	0.39	0.41	100
16	0.62	0.44	0.51	100
17	0.50	0.65	0.57	100
18	0.36	0.38	0.37	100
19	0.38	0.37	0.38	100
20	0.68	0.77	0.72	100
21	0.44	0.64	0.52	100
22	0.49	0.45	0.47	100
23	0.70	0.61	0.65	100
24	0.72	0.62	0.67	100
25	0.36	0.38	0.37	100
26	0.48	0.30	0.37	100
27	0.29	0.25	0.27	100
28	0.74	0.70	0.72	100
29	0.45	0.47	0.46	100
30	0.46	0.43	0.45	100
31	0.39	0.43	0.41	100
32	0.39	0.28	0.33	100
33	0.51	0.43	0.47	100
34	0.44	0.43	0.44	100
35	0.27	0.32	0.29	100
36	0.46	0.47	0.47	100
37	0.48	0.51	0.50	100
38	0.28	0.30	0.29	100
39	0.59	0.55	0.57	100
40	0.40	0.42	0.41	100
41	0.68	0.69	0.69	100
42	0.49	0.49	0.49	100
43	0.61	0.49	0.54	100
44	0.17	0.14	0.15	100
45	0.38	0.34	0.36	100
46	0.26	0.36	0.30	100
47	0.58	0.57	0.58	100
48	0.63	0.85	0.72	100
49	0.59	0.50	0.54	100
50	0.31	0.26	0.28	100
51	0.34	0.39	0.36	100
52	0.54	0.72	0.62	100
53	0.55	0.77	0.64	100

54	0.48	0.65	0.55	100
55	0.20	0.16	0.18	100
56	0.62	0.65	0.63	100
57	0.57	0.54	0.55	100
58	0.63	0.64	0.64	100
59	0.53	0.37	0.44	100
60	0.63	0.79	0.70	100
61	0.50	0.52	0.51	100
62	0.51	0.62	0.56	100
63	0.51	0.44	0.47	100
64	0.31	0.31	0.31	100
65	0.23	0.28	0.25	100
66	0.41	0.37	0.39	100
67	0.44	0.23	0.30	100
68	0.75	0.86	0.80	100
69	0.50	0.66	0.57	100
70	0.45	0.50	0.47	100
71	0.64	0.43	0.51	100
72	0.20	0.16	0.18	100
73	0.38	0.43	0.40	100
74	0.39	0.20	0.26	100
75	0.61	0.70	0.65	100
76	0.55	0.80	0.65	100
77	0.32	0.24	0.27	100
78	0.30	0.29	0.29	100
79	0.49	0.43	0.46	100
80	0.33	0.22	0.26	100
81	0.60	0.47	0.53	100
82	0.61	0.76	0.68	100
83	0.41	0.38	0.40	100
84	0.42	0.32	0.36	100
85	0.60	0.58	0.59	100
86	0.39	0.61	0.47	100
87	0.47	0.60	0.52	100
88	0.57	0.52	0.54	100
89	0.56	0.44	0.49	100
90	0.41	0.48	0.44	100
91	0.64	0.56	0.60	100
92	0.31	0.47	0.38	100
93	0.30	0.19	0.23	100
94	0.74	0.84	0.79	100
95	0.49	0.55	0.52	100
96	0.48	0.35	0.40	100
97	0.36	0.53	0.43	100
98	0.22	0.22	0.22	100
99	0.47	0.51	0.49	100
accuracy			0.47	10000

Accuracy report for Model 2:

	precision	recall	f1-score	support
0	0.73	0.87	0.79	100

1	0.66	0.72	0.69	100
2	0.52	0.49	0.50	100
3	0.31	0.38	0.34	100
4	0.30	0.54	0.39	100
5	0.58	0.59	0.59	100
6	0.69	0.74	0.71	100
7	0.66	0.55	0.60	100
8	0.80	0.82	0.81	100
9	0.80	0.70	0.74	100
10	0.54	0.48	0.51	100
11	0.42	0.40	0.41	100
12	0.64	0.69	0.67	100
13	0.84	0.56	0.67	100
14	0.72	0.46	0.56	100
15	0.68	0.63	0.66	100
16	0.71	0.62	0.66	100
17	0.80	0.78	0.79	100
18	0.60	0.44	0.51	100
19	0.59	0.50	0.54	100
20	0.89	0.79	0.84	100
21	0.64	0.84	0.73	100
22	0.61	0.60	0.60	100
23	0.69	0.86	0.76	100
24	0.71	0.77	0.74	100
25	0.53	0.55	0.54	100
26	0.57	0.56	0.57	100
27	0.34	0.57	0.43	100
28	0.87	0.76	0.81	100
29	0.72	0.54	0.62	100
30	0.48	0.65	0.55	100
31	0.66	0.61	0.64	100
32	0.58	0.59	0.58	100
33	0.64	0.59	0.61	100
34	0.67	0.63	0.65	100
35	0.42	0.33	0.37	100
36	0.72	0.56	0.63	100
37	0.62	0.65	0.64	100
38	0.58	0.45	0.51	100
39	0.84	0.76	0.80	100
40	0.64	0.47	0.54	100
41	0.84	0.79	0.81	100
42	0.63	0.66	0.64	100
43	0.65	0.71	0.68	100
44	0.33	0.41	0.36	100
45	0.43	0.62	0.51	100
46	0.43	0.47	0.45	100
47	0.66	0.57	0.61	100
48	0.83	0.86	0.85	100
49	0.73	0.85	0.79	100
50	0.33	0.34	0.34	100
51	0.72	0.59	0.65	100
52	0.54	0.80	0.65	100
53	0.70	0.85	0.77	100
54	0.62	0.74	0.68	100

55	0.25	0.22	0.23	100
56	0.89	0.81	0.85	100
57	0.77	0.66	0.71	100
58	0.75	0.79	0.77	100
59	0.69	0.49	0.57	100
60	0.89	0.72	0.80	100
61	0.73	0.70	0.71	100
62	0.69	0.66	0.68	100
63	0.70	0.50	0.58	100
64	0.38	0.43	0.41	100
65	0.55	0.41	0.47	100
66	0.72	0.58	0.64	100
67	0.45	0.53	0.48	100
68	0.87	0.91	0.89	100
69	0.78	0.76	0.77	100
70	0.69	0.64	0.66	100
71	0.70	0.81	0.75	100
72	0.23	0.41	0.29	100
73	0.40	0.43	0.41	100
74	0.42	0.45	0.44	100
75	0.92	0.80	0.86	100
76	0.85	0.79	0.82	100
77	0.64	0.53	0.58	100
78	0.49	0.48	0.49	100
79	0.72	0.57	0.64	100
80	0.34	0.38	0.36	100
81	0.66	0.73	0.70	100
82	0.93	0.84	0.88	100
83	0.70	0.50	0.58	100
84	0.60	0.57	0.58	100
85	0.68	0.83	0.75	100
86	0.71	0.67	0.69	100
87	0.70	0.71	0.71	100
88	0.82	0.58	0.68	100
89	0.76	0.71	0.73	100
90	0.65	0.80	0.72	100
91	0.74	0.69	0.72	100
92	0.55	0.62	0.58	100
93	0.43	0.46	0.45	100
94	0.87	0.87	0.87	100
95	0.61	0.54	0.57	100
96	0.53	0.54	0.54	100
97	0.67	0.62	0.64	100
98	0.40	0.36	0.38	100
99	0.67	0.62	0.64	100
accuracy			0.62	10000

Accuracy report for Model 3:

	precision	recall	f1-score	support
0	0.89	0.88	0.88	100
1	0.87	0.76	0.81	100

2	0.67	0.60	0.63	100
3	0.59	0.41	0.49	100
4	0.47	0.48	0.47	100
5	0.56	0.74	0.64	100
6	0.69	0.79	0.73	100
7	0.77	0.63	0.69	100
8	0.75	0.83	0.79	100
9	0.84	0.82	0.83	100
10	0.47	0.50	0.49	100
11	0.50	0.51	0.51	100
12	0.71	0.78	0.74	100
13	0.75	0.65	0.70	100
14	0.53	0.73	0.62	100
15	0.69	0.65	0.67	100
16	0.74	0.73	0.73	100
17	0.76	0.81	0.79	100
18	0.62	0.64	0.63	100
19	0.68	0.62	0.65	100
20	0.88	0.85	0.86	100
21	0.88	0.76	0.82	100
22	0.59	0.73	0.65	100
23	0.81	0.76	0.78	100
24	0.87	0.76	0.81	100
25	0.59	0.54	0.57	100
26	0.51	0.79	0.62	100
27	0.47	0.54	0.50	100
28	0.76	0.79	0.77	100
29	0.60	0.66	0.63	100
30	0.68	0.58	0.63	100
31	0.81	0.55	0.65	100
32	0.80	0.52	0.63	100
33	0.59	0.71	0.64	100
34	0.77	0.76	0.76	100
35	0.43	0.49	0.46	100
36	0.87	0.74	0.80	100
37	0.77	0.69	0.73	100
38	0.69	0.46	0.55	100
39	0.56	0.86	0.68	100
40	0.87	0.58	0.69	100
41	0.86	0.79	0.82	100
42	0.39	0.83	0.53	100
43	0.80	0.72	0.76	100
44	0.34	0.38	0.36	100
45	0.49	0.64	0.56	100
46	0.51	0.51	0.51	100
47	0.49	0.74	0.59	100
48	0.80	0.94	0.86	100
49	0.74	0.84	0.79	100
50	0.55	0.43	0.48	100
51	0.77	0.68	0.72	100
52	0.67	0.48	0.56	100
53	0.75	0.95	0.84	100
54	0.85	0.76	0.80	100
55	0.47	0.24	0.32	100

56	0.93	0.85	0.89	100
57	0.74	0.78	0.76	100
58	0.82	0.88	0.85	100
59	0.63	0.68	0.65	100
60	0.75	0.88	0.81	100
61	0.59	0.78	0.67	100
62	0.88	0.63	0.73	100
63	0.38	0.67	0.49	100
64	0.63	0.51	0.56	100
65	0.63	0.49	0.55	100
66	0.71	0.79	0.75	100
67	0.61	0.48	0.54	100
68	0.89	0.93	0.91	100
69	0.84	0.81	0.83	100
70	0.77	0.75	0.76	100
71	0.83	0.63	0.72	100
72	0.44	0.35	0.39	100
73	0.59	0.50	0.54	100
74	0.39	0.51	0.44	100
75	0.90	0.77	0.83	100
76	0.92	0.82	0.87	100
77	0.59	0.62	0.60	100
78	0.41	0.67	0.51	100
79	0.63	0.71	0.67	100
80	0.66	0.42	0.51	100
81	0.67	0.72	0.69	100
82	0.92	0.88	0.90	100
83	0.85	0.52	0.65	100
84	0.80	0.63	0.70	100
85	0.87	0.80	0.83	100
86	0.65	0.73	0.69	100
87	0.71	0.77	0.74	100
88	0.57	0.86	0.69	100
89	0.74	0.78	0.76	100
90	0.82	0.77	0.79	100
91	0.84	0.72	0.77	100
92	0.78	0.58	0.67	100
93	0.53	0.49	0.51	100
94	0.90	0.87	0.88	100
95	0.68	0.69	0.69	100
96	0.65	0.37	0.47	100
97	0.81	0.61	0.70	100
98	0.57	0.41	0.48	100
99	0.74	0.57	0.64	100
accuracy			0.67	10000

In [93]:
```
### Confusion matrix
```

Confusion matrix

In [92]:
```
#labels = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Sh
```

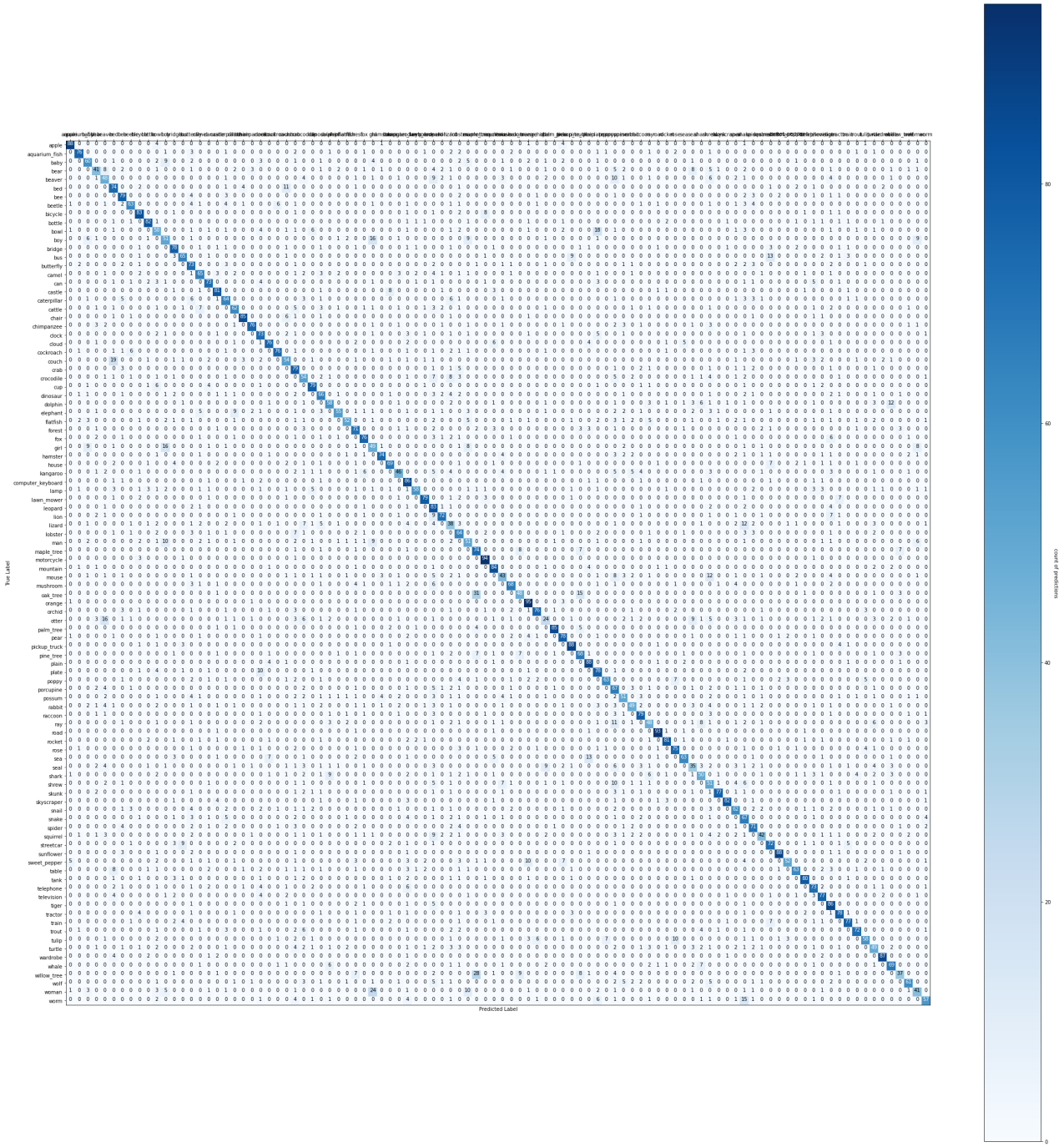
```
# Convert predictions classes to one hot vectors
y_pred_classes = np.argmax(pred, axis=1)
# Convert validation observations to one hot vectors
y_true = np.argmax(y_test, axis=1)
# Errors are difference between predicted labels and true labels
errors = (y_pred_classes - y_true != 0)

y_pred_classes_errors = y_pred_classes[errors]
y_pred_errors = pred[errors]
y_true_errors = y_true[errors]
X_test_errors = X_test[errors]

cm = confusion_matrix(y_true, y_pred_classes)
thresh = cm.max() / 2.

fig, ax = plt.subplots(figsize=(32,32))
im, cbar = heatmap(cm, labels, labels, ax=ax,
                   cmap=plt.cm.Blues, cbarlabel="count of predictions")
texts = annotate_heatmap(im, data=cm, threshold=thresh)

fig.tight_layout()
plt.show()
```

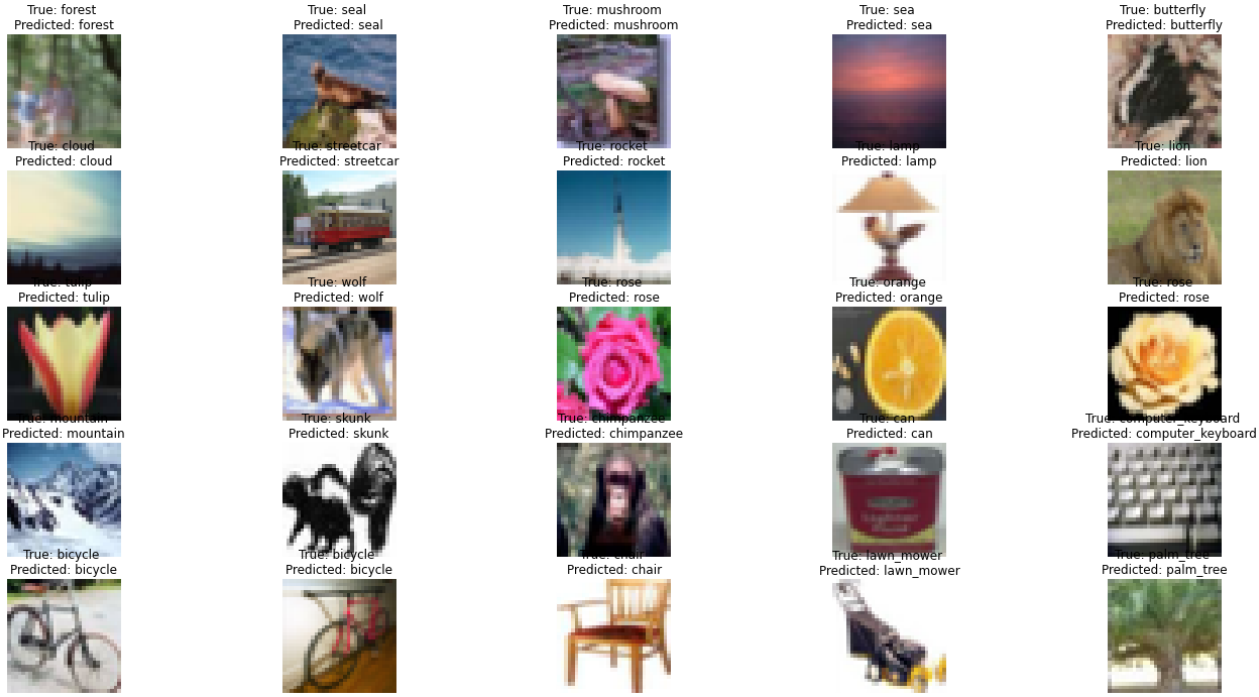


Examples of correct classifications for Model 3.:

In [90]:

```
row = 5
column = 5
fig, axes = plt.subplots(row, column, figsize=(22,12))
axes = axes.ravel()

classified_idx = np.where(y_pred_classes == y_true)[0]
for i in np.arange(0, row*column):
    axes[i].imshow(X_test[classified_idx[i]])
    axes[i].set_title("True: %s \nPredicted: %s" % (labels[y_true[classified_idx[i]]],
                                                   labels[y_pred_classes[classified_idx[
    axes[i].axis('off')
    plt.subplots_adjust(wspace=1)
```

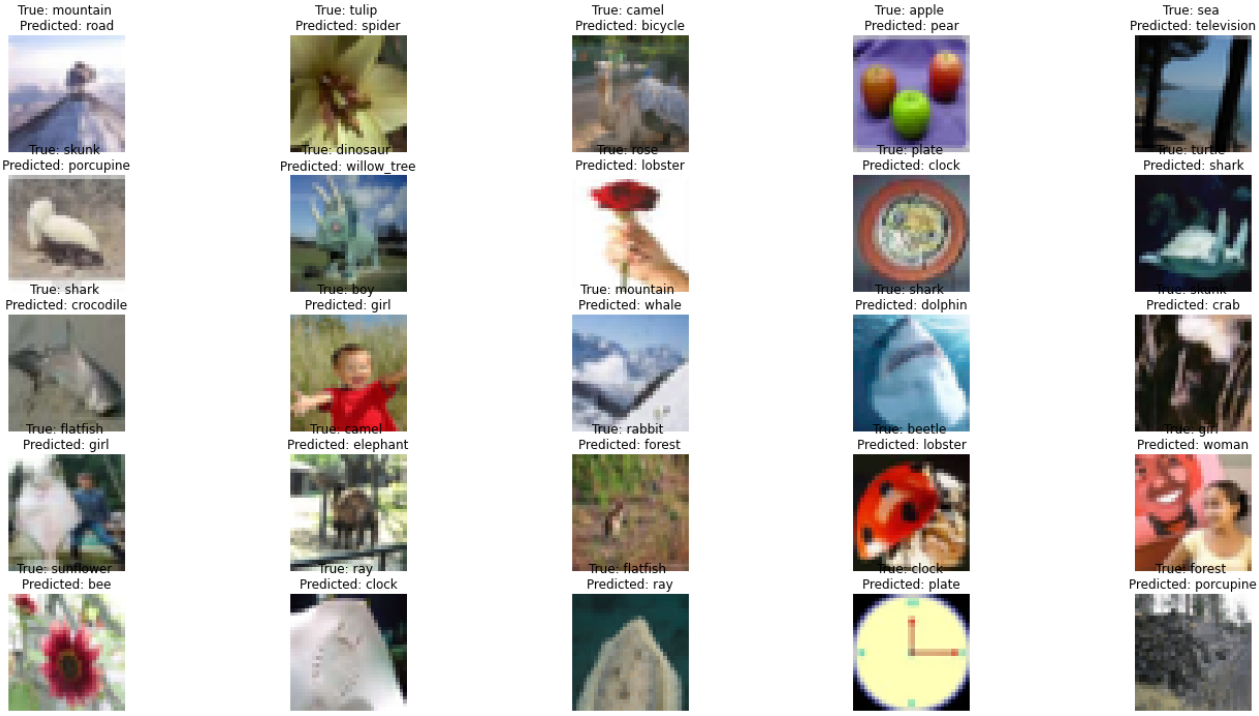


Examples of misclassifications for Model 3.:

In [91]:

```
row = 5
column = 5
fig, axes = plt.subplots(row, column, figsize=(22,12))
axes = axes.ravel()

misclassified_idx = np.where(y_pred_classes != y_true)[0]
for i in np.arange(0, row*column):
    axes[i].imshow(X_test[misclassified_idx[i]])
    axes[i].set_title("True: %s \nPredicted: %s" % (labels[y_true[misclassified_idx[i]]]
                                                   labels[y_pred_classes[misclassified_i
    axes[i].axis('off')
    plt.subplots_adjust(wspace=1)
```



5 A paragraph explaining which of your Deep Learning models you recommend as a final model that best fits your needs in terms of accuracy or explainability.

Due to characteristics of the problem I value accuracy more than explainability.
The model I recommend as a final one is the model with highest accuracy on validation data set.
That is Model 3. which uses some Image Augmentation methods.

6 Summary Key Findings and Insights, which walks your reader through the main findings of your modeling exercise.

All train models had more less similar depth of initial architecture. Nonetheless they were slightly diffrent in few aspects.

- Model 1 was trained without Pooling.
  - in a comparison to Model 2 significant drop of accuracy was noted
  - removing Pooling increased significantly overall time needed for processing of each Epoch
- Model 2 was trained with same architecture as Model 1 but with an addition of Pooling
  - adding Pooling lowered time needed for processing of each Epoch
  - the accuracy increased from X to Y
- Model 3 was trained with same architecture as Model 2 but with an addition of Image Augmentation
  - the accuracy increased from Y to Z

7 Suggestions for next steps in analyzing this data, which may include suggesting revisiting this model or adding specific data features to achieve a better model.

There is still a space for more experiments with Image Augmentation methods. There are still available options not explored on that matter.
Nonetheless the question is: How much more accuracy is there to be dig out from initial set of images?

Moving from Keras which is indeed high level library to more sophisticated one like TensorFlow might prove beneficial as well.