```
<div id="dataframecontent" style="overflow-x:auto"></div>

<script>

const url = "https://storage.googleapis.com/tfds-data/visualization/dataframe/diabetic_retinopathy_detection-250K-3.0.0.html";

const dataButton = document.getElementById('displaydataframe');

dataButton.addEventListener('click', async () => {

  // Disable the button after clicking (dataframe loaded only once).

  dataButton.disabled = true;


  const contentPane = document.getElementById('dataframecontent');

  try {

    const response = await fetch(url);

    // Error response codes don't throw an error, so force an error to show

    // the error message.

    if (!response.ok) throw Error(response.statusText);


    const data = await response.text();

    contentPane.innerHTML = data;

  } catch (e) {

    contentPane.innerHTML =

        'Error loading examples. If the error persist, please open '

        + 'a new issue.';

  }

});

</script>


{% endframebox %}


<!-- mdformat on -->


## diabetic_retinopathy_detection/btgraham-300
```

Detecting diabetic retinopathy (DR) using convolutional neural networks (CNNs) is a significant area of research in medical image analysis. Here's a basic overview of how you could approach this task:

*** Data Collection***
Gather a large dataset of retinal images, preferably labeled by experts to indicate the presence and severity of DR.

Preprocessing
Preprocess the images to enhance features and ensure uniformity. This may include resizing, normalization, and contrast adjustment.

Data Augmentation
Since medical datasets are often limited in size, data augmentation techniques such as rotation, flipping, and scaling can be applied to artificially increase the size of the dataset and improve the model's robustness.

Model Selection
Choose a suitable CNN architecture. Common choices include VGG, ResNet, Inception, or custom architectures tailored to the task.

Transfer Learning
Pre-train the chosen CNN on a large dataset (e.g., ImageNet) and fine-tune it on your retinal image dataset. This helps the model to learn general features from the pre-training and adapt them to the specific task of DR detection.

* **Config description**: Images have been preprocessed as the winner of the
  Kaggle competition did in 2015: first they are resized so that the radius of
  an eyeball is 300 pixels, then they are cropped to 90% of the radius, and
  finally they are encoded with 72 JPEG quality.

* **Dataset size**: `3.65 GiB`

The size of the dataset required for training a diabetic retinopathy detection model using CNNs can vary depending on several factors, including the complexity of the problem, the diversity of the data, and the chosen architecture. However, generally, a larger dataset is beneficial for training more accurate and robust models.For medical imaging tasks like diabetic retinopathy detection, it's ideal to have thousands to tens of thousands of labeled images for training. This allows the model to learn various features and patterns associated with different stages of the disease. Additionally, having a sizable dataset helps prevent overfitting and improves generalization to unseen data.It's essential to ensure that the dataset is diverse and representative of the target population, encompassing different demographics, disease severities, and imaging conditions. Data augmentation techniques can also be employed to artificially increase the effective size of the dataset, especially if the original dataset is limited.

* **Figure**

Training and Evaluation:

- Describe how you split your dataset into training, validation, and test sets.

- Provide details about hyperparameter tuning, loss functions, and optimization techniques.

- Include evaluation metrics (e.g., accuracy, precision, recall) and explain why you chose them.

Results and Interpretation:

- Present the performance of your model on the test set. Include confusion matrices or ROC curves.

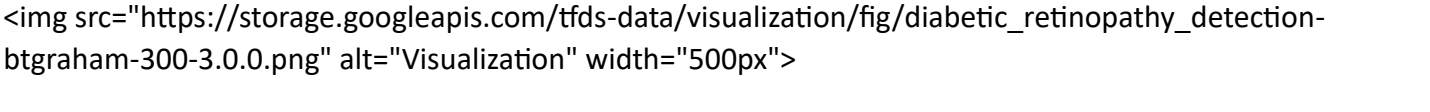
- Discuss any limitations or challenges you encountered during training and testing.

Interpretability and Explainability:

- If you used deep learning models, consider explaining their predictions using techniques like Grad-CAM or LIME.

- Highlight areas where the model struggled and potential reasons for misclassifications.

([tfds.show_examples](https://www.tensorflow.org/datasets/api_docs/python/tfds/visualization/show_examples)):



* **Examples**

([tfds.as_dataframe](https://www.tensorflow.org/datasets/api_docs/python/tfds/as_dataframe)):

<!-- mdformat off(HTML should not be auto-formatted) -->

{% framebox %}

<button id="displaydataframe">Display examples...</button>

<script>

const url = "https://storage.googleapis.com/tfds-data/visualization/dataframe/diabetic_retinopathy_detection-btgraham-300-3.0.0.html";

const dataButton = document.getElementById('displaydataframe');

dataButton.addEventListener('click', async () => {

// Disable the button after clicking (dataframe loaded only once).

dataButton.disabled = true;

```
  const contentPane = document.getElementById('dataframecontent');
  try {
    const response = await fetch(url);
    // Error response codes don't throw an error, so force an error to show
    // the error message.
    if (!response.ok) throw Error(response.statusText);

    const data = await response.text();
    contentPane.innerHTML = data;
  } catch (e) {
    contentPane.innerHTML =
        'Error loading examples. If the error persist, please open '
        + 'a new issue.';
  }
});
</script>

{% endframebox %}

<!-- mdformat on -->
```