## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the

## Training and targets

Targets for R-CNN are calculated in almost the same way as the RPN targets, but taking into account the different possible classes. We take the proposals and the ground-truth boxes, and calculate the IoU between them.

Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth. Those that have between 0.1 and 0.5 get labeled as background. Contrary to what we did while assembling targets for the RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals and we are more interested in solving the harder cases. Of course, all these values are hyperparameters that can be tuned to better fit the type of objects that you are trying to find.

The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.

We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.

Following the same path as we did for the RPNs losses, the classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box. We have to be careful when getting that loss since the output of the R-CNN fully connected network for bounding box regressions has one prediction for each of the classes. When calculating the loss, we only have to take into account the one for the correct class.

## Post processing

Similar to the RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them.

In order to apply the bounding box adjustments we have to take into account which is the class with the highest probability for that proposal. We also have to ignore those proposals that have the background class as the one with the highest probability.

After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the